CHOISIR L'AGILITÉ

Du développement logiciel à la gouvernance

Mathieu Boisvert
Conseiller en adoption des méthodes agiles, Scrum Master

Sylvie Trudel
Spécialiste en développement logiciel et amélioration des processus, Scrum Master

DUNOD

Illustration de couverture :
Plage des seychelles © Pat on stock - Fotolia.com

ISBN 978-2-10-056874-1

Table des matières

Avant-propos

CE QUI NOUS A AMENÉS À ÉCRIRE CE LIVRE

Les méthodes agiles sont dites légères, car elles sont moins prescriptives que celles dites traditionnelles. Elles sont fondées sur un manifeste composé de quatre valeurs et douze principes et elles ne définissent que très peu de rôles, de livrables et de pratiques (Figure 1[1]).

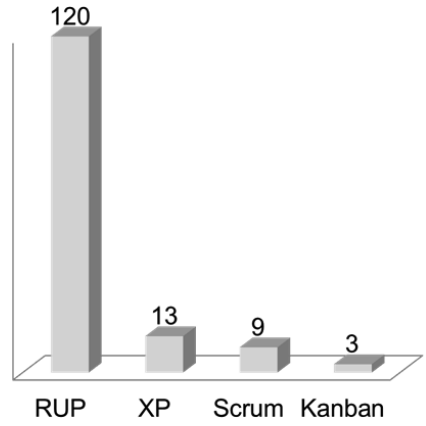


Figure 1 — Comparaison du niveau prescriptif des méthodes, basé sur le nombre de rôles, de livrables et pratiques définis par méthode

Plusieurs praticiens apprécient le caractère léger des méthodes agiles, car elles s'adaptent bien aux différents contextes des projets de développement logiciel. En 2004, après trois années comme développeur dans une institution bancaire française, Mathieu a découvert la méthode Scrum. Il a apprécié les méthodes agiles qui lui ont permis de se concentrer sur la réalisation de produits de qualité, avec une

1. Tiré de l'ouvrage *Kanban and Scrum - making the most of both* d'Henrik Kniberg et Mattias Skarin, http://www.infoq.com/minibooks/kanban-scrum-minibook, consulté en décembre 2009.

cadence de livraison de plus en plus prévisible, en collaboration avec son équipe de développement. D'équipes en équipes, il a cependant découvert les limites du travail au sein d'un noyau agile dans une entreprise qui ne l'est pas. Devenu Scrum Master, Mathieu s'est intéressé aux autres méthodes et à l'approche agile en général, ce qui lui a permis d'aider son équipe à franchir ces frontières et de sensibiliser les organisations aux bienfaits des méthodes agiles. Son expérience et son succès auprès de différentes organisations et plusieurs organismes l'amène aujourd'hui à conseiller au démarrage et à la transition aux méthodes agiles.

Le parcours de Sylvie diffère de celui de Mathieu, forte de ses 26 ans d'expérience en développement logiciel. Après avoir été programmeur-analyste, analyste fonctionnel et architecte fonctionnel pendant onze ans, elle a fait un virage vers les processus de développement et de maintenance logiciel au milieu des années quatre-vingt-dix, dans une industrie où le CMM (*Capability Maturity Model for Software*) omniprésent allait devenir un modèle à suivre *de facto*. Au début des années 2000, Sylvie a accompagné des organisations, petites et grandes, à améliorer leur processus de développement et à en mesurer la performance sous forme de coût unitaire, en heures par point de fonction COSMIC (*Common Software Measurement International Consortium*, http://www.cosmicon.com). En parallèle, elle a obtenu sa certification de Scrum Master qu'elle utilise en industrie pour transformer les façons de faire du logiciel. En appliquant l'agilité, plusieurs des organisations qu'elle a conseillées ont réussi à réduire de moitié leur coût unitaire moyen (mesuré objectivement), comparativement au coût unitaire moyen de leur méthodologie traditionnelle. Elle a été la première, en septembre 2006, à présenter le mariage des pratiques agiles et du CMMI lors d'un séminaire du SPIN de Montréal (*Software Process Improvement Network*), puis à la Fédération de l'Informatique du Québec (maintenant appelé le *Réseau Action TI*) le mois suivant, en collaboration avec François Beauregard, le premier instructeur Scrum francophone certifié.

Les avantages proposés par les méthodes dites « légères » plaisent aux gestionnaires et les clients que nous avons rencontrés se sont souvent montrés collaboratifs pour modifier leur cadre méthodologique de développement traditionnel.

La plupart des clients que nous avons conseillés ont été réceptifs aux méthodes légères. Ils ont accepté de modifier leur processus de développement pour profiter des avantages offerts par les méthodes agiles :

- qualité améliorée ;
- délais de mise en marché réduits ;
- engagements tenus par les équipes ;
- productivité améliorée ;
- et plusieurs autres avantages.

Aujourd'hui, le marché est friand de l'agilité et les mandats d'accompagnement sont plus ambitieux. Nous accompagnons maintenant des organisations à faire une transition agile complète : du coaching jusqu'à la rédaction du cadre méthodologique. Au moment d'écrire ce livre, Mathieu constate l'étendue des besoins dans une organisation d'envergure où les gestionnaires et les membres d'équipes doivent

développer, comprendre et communiquer de nouveaux processus. Sylvie le faisait depuis plus de quinze ans de façon générale et depuis cinq ans dans un contexte de transition agile. Jusqu'à maintenant, nos clients étaient bons joueurs et conservaient la responsabilité d'adapter leurs processus organisationnels.

Nous avions compris depuis longtemps que les méthodes agiles n'étaient pas vraiment des méthodes, mais qu'elles sont plutôt des philosophies de travail demandant énormément de rigueur pour être efficaces. Cependant, pour changer la culture d'une entreprise de quelques centaines d'employés, la rigueur ne suffit pas. Les personnes impliquées ont besoin d'un cadre pour évoluer dans une organisation où chacun se demandera : Quel est mon rôle ? Quelles sont mes responsabilités ? Qui consomme mes livrables ? Qui sont mes parties prenantes ?

Les individus sont prêts à être agiles lorsqu'ils sont informés. C'est une erreur que de répondre à ces questions en utilisant seulement la littérature agile. Lors d'une d'intervention en clientèle, nous, Sylvie, Mathieu et Elsa Larreur, une collègue et Scrum Master d'expérience, avons eu l'opportunité d'associer le langage traditionnel ainsi que les principes des processus organisationnels avec les concepts des approches agiles. Nous avons traduit les responsabilités des rôles de Scrum et des membres d'une organisation avec un RACI (Responsable, Approbateur, Contributeur, tenu Informé). Nous avons expliqué le processus de l'amélioration continue agile à l'aide d'un modèle de gestion de la qualité PDCA (*Plan-Do-Check-Act*). Nous avons démontré que les points d'effort ne sont pas les seules façons d'estimer l'ampleur d'un projet et que le CMMI (*Capability Maturity Model Integration*) n'est pas l'ennemi des méthodes agiles.

La littérature traitant des méthodes agiles donne à tort l'impression qu'elles ne sont pas compatibles avec les processus déjà présents dans les organisations. Au-delà d'être persuadé des bienfaits de l'agilité, vous devez savoir qu'il y aura des impacts certains dans TOUS les niveaux de l'organisation. Cela aura pour effet d'amener votre organisation à revoir son cadre méthodologique (les processus organisationnels).

TERMINOLOGIE : COMMENT S'Y RETROUVER ?

Exemple de conversation entendue lors de nos interventions :

Alain : T'es au courant du nouveau truc qu'on va mettre en place ? C'est la méthode « agile ».

Marie-Jeanne : « agile », ce n'est pas une méthode, c'est une approche. Veux-tu parler des méthodes Scrum et XP ?

Alain : Oui c'est ça, Scrum et XP. Et en plus, il parait qu'on poursuit avec la démarche « CMMI ». Je me demande comment on va arrimer toutes ces méthodes. Le CMMI, c'est une méthode, non ?

MJ : Non, c'est un modèle.

Alain : Ah bon ! ? Est-ce la même chose qu'une norme ?

MJ : Non, ce n'est pas une norme. Il contient des objectifs et des pratiques.

Alain : Est-ce un ensemble de techniques ?

MJ : Non, c'est plutôt un ensemble de pratiques reconnues comme étant les « meilleures ».
Alain : Ah, mais c'est un processus !
MJ : Non plus. Un processus est caractérisé par des flux d'opérations, un système d'activités qui utilise des ressources pour transformer des éléments d'entrée en éléments de sortie[1].
Alain : Mais n'est-ce pas ce qu'on appelle une « procédure » ?
MJ : Non, car une procédure est une description pas à pas d'une opération ou d'une partie d'opération.
Alain : Dans ce cas, parle-t-on de méthodologie ?
MJ : Pas du tout. Une méthodologie est une démarche suivie, une manière de procéder[2].
Alain : Qu'est-ce qu'un modèle alors ?

Avant de vous lancer dans votre lecture, nous aimerions vous inviter à revoir la définition de certains termes utilisés dans le présent ouvrage. La confusion des termes et les questionnements du précédent dialogue sont fréquents dans les organisations qui démarrent une démarche d'amélioration de processus. C'est pourquoi il importe de préciser chacun des termes pour éviter toute confusion, non pas à partir d'une simple définition trouvée dans un dictionnaire mais expliqué de manière concrète et étoffé d'exemples. Pour en faciliter la lecture, nous les avons décrits ci-après en ordre alphabétique.

Approche

Une approche est « *une manière d'aborder une question, un problème* »[3], ce qui est généralement guidé par des principes et des valeurs. Une approche ne précise pas « quoi » faire, ni « comment faire ». Donc, quand on parle de « l'agilité » ou « d'agile », on parle alors d'approche de l'agilité ou d'approche agile. Une approche pourra être mise en œuvre par des méthodes qui respectent les principes et les valeurs qui définissent cette approche.

Méthode

Une méthode est une façon de faire. Elle décrit « *comment* » on fait les choses. En général, une méthode traite d'un aspect précis faisant partie d'un processus plus large ou d'une méthodologie. Voici quelques exemples :

- une méthode de gestion de projet et d'exigences (ex. Scrum) ;
- une méthode de définition d'exigences via des tests unitaires (ex. TDD) ;

1. Source : Norme ISO 9000:2000, Systèmes de management de la qualité – Principes essentiels et vocabulaire, Organisation Internationale de Normalisation, 2000.
2. Source : http://www.linternaute.com/dictionnaire/fr/definition/methodologie/, consulté en juillet 2010.
3. Source : http://www.linternaute.com/dictionnaire/fr/definition/approche/ , consulté en juillet 2010.

- une méthode d'estimation de projet par consensus (ex. Poker planning) ;
- une méthode de revue de code en continu (ex. programmation par paire, appelée aussi binômage) ;
- une méthode de restructuration de code.

Ces exemples de méthodes traitent donc d'une opération précise ou d'un ensemble d'opérations pouvant faire partie d'un processus de développement et d'évolution du logiciel. Synonyme de « *technique* ».

Méthodologie

On trouve plusieurs définitions du mot « *méthodologie* » sur Internet et dans les dictionnaires, mais celle qui ressemble le plus à l'utilisation qu'on en fait dans le contexte du développement logiciel est un « *ensemble des méthodes appliquées à un domaine particulier* »[1].

Donc, un ensemble de méthodes que l'on peut agencer selon nos besoins organisationnels ou de projet pour former un processus. Une méthodologie pourrait s'apparenter à un menu de restaurant : nous utiliserons plusieurs items de son contenu mais pas forcément l'entièreté. Par exemple, une méthodologie pourrait fournir diverses méthodes pour documenter les exigences : cas d'utilisation, scénarios utilisateurs (*user stories*), méthodes formelles, ou langage naturel. Cependant, au démarrage d'un projet, il faudra décider laquelle de ces méthodes sera celle adoptée par le projet puisqu'il est souvent peu probable qu'un projet en utilise plusieurs. C'est souvent le cas à l'échelle organisationnelle, alors qu'on préfère limiter le nombre de méthodes à utiliser de façon à pouvoir réduire la courbe d'apprentissage des intervenants participant à divers projets.

Certaines méthodologies ont été ou sont populaires : Merise (à une certaine époque), Macroscope (surtout au Québec), *Rational Unified Process* (RUP) qui, contrairement à ce que son nom laisse penser, n'est pas un processus mais une méthodologie.

Modèle

Un modèle est une « *représentation théorique d'un système d'éléments et de relations plus ou moins complexes*[2] ». En développement logiciel, on parle de modèle de « meilleures pratiques » tel que le CMMI, mais aussi de modèle de données, modèle objets, et modèle de traitements qui sont des représentations d'un aspect du logiciel qu'ils décrivent.

> « *Tous les modèles sont faux, quelques-uns sont utiles.* »
> Georges Box[3]

1. Source : http://dictionnaire.reverso.net/francais-definition/méthodologie , consulté en juillet 2010.
2. Source : http://www.mediadico.com/dictionnaire/definition/Modele/1 , consulté en juillet 2010.
3. Traduction libre de : http://en.wikiquote.org/wiki/George_Box, consulté en février 2011.

Un modèle est une représentation simplifiée qui permet d'agir sans s'empêtrer dans de trop nombreux détails, donc il n'est jamais exhaustif ni complet, d'où leur qualificatif de « faux », mais souvent utile.

Norme

Une norme est un « *document établi par consensus, qui fournit, pour des usages communs et répétés, des règles, des lignes directrices ou des caractéristiques, pour des activités ou leurs résultats, garantissant un niveau d'ordre optimal dans un contexte donné* »[1]. Les normes sont publiées par des organismes de normalisation et sont généralement utilisées de façon volontaire.

Pratique

Une pratique est une « *activité volontaire ayant pour but d'obtenir des résultats concrets en appliquant des règles ou des principes* »[2]. Dans le CMMI, chacune des pratiques recommandées est directement liée à l'atteinte d'un objectif. Souvent accolé au mot « bonne » ou « meilleure », en anglais « *best* », ce qui évoque la capitalisation, un élément vital de modèles comme le CMMI. Dans les méthodes agiles, nous verrons les pratiques qui appliquent les principes et les valeurs de l'agilité.

Procédure

Une procédure est une marche à suivre, une description pas à pas d'une série d'opérations, généralement effectuées par le même individu. Une procédure peut faire appel à une méthode, une technique ou une pratique pour une opération particulière.

Processus

Un processus est une série d'étapes qui transforme des intrants (*inputs*) en extrants (*outputs*) et au cours desquelles on mesurera certains paramètres (ex. effort, durée, quantité d'*inputs* ou d'*outputs*, taille, qualité). Plus englobant qu'une procédure pas à pas, le processus est conçu pour être exécuté par plus d'un individu, voire plusieurs groupes. On représente souvent un processus sous forme de « *workflow* » (flux de travail). Un processus couvre par exemple un ensemble de procédures : le processus de démarrage de projets peut référer aux procédures d'estimation, de définition de la portée, d'organisation d'équipe, etc.

Technique

Une technique est un « *ensemble des procédés mis en œuvre dans un métier, un art, une science* » ou synonyme de « méthode, moyen, manière de faire »[3].

1. Source : Guide ISO/CEI 2.
2. Source : http://dictionnaire.sensagent.com/pratique/fr-fr/ , consulté en août 2010.
3. Source : http://www.linternaute.com/dictionnaire/fr/definition/technique/, consulté en août 2010.

COMMENT UTILISER CE LIVRE ?

Aspects concernés par la transition à l'agilité

Au cours d'une transition à l'agilité, plusieurs aspects des processus d'une organisation sont concernés ou impactés et ce, quelle que soit la taille des organisations. Nous avons regroupé ces aspects en quatre niveaux concentriques :

1. les projets ;
2. l'encadrement des projets ;
3. les processus et les individus ;
4. la gouvernance.

Processus de premier niveau : les projets

Les projets représentent le premier aspect concerné par une transition des processus à l'agilité (Figure 2). Une organisation commence par un ou deux projets pilotes pour mieux comprendre ce qu'une transition implique dans le reste de l'organisation à partir du moment où le choix de l'agilité est confirmé.

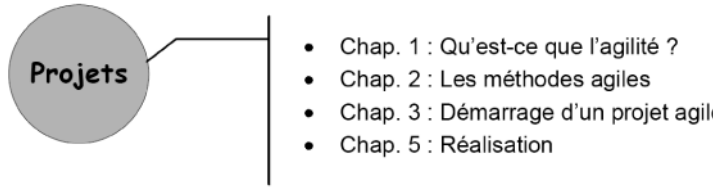


Figure 2 — Les projets comme premier aspect concerné par une transition agile.

Processus de deuxième niveau : l'encadrement des projets

Au cours des premiers projets en mode agile, les intervenants du bureau de projets risquent de devoir adapter la collecte des données d'avancement des travaux afin d'harmoniser les pratiques de reddition de comptes pour tous les projets, qu'ils soient exécutés en mode agile ou traditionnel. La gestion de projet est un deuxième aspect concerné par l'adoption de l'agilité (Figure 3).

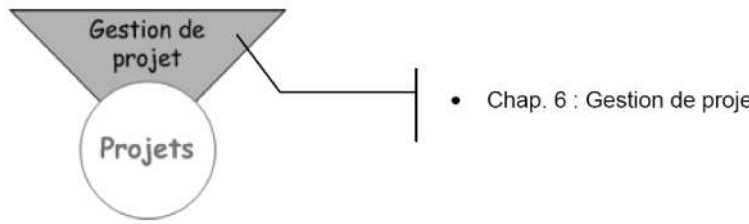


Figure 3 — L'aspect de la gestion de projet concerné par une transition agile.

En parallèle, les équipes qui avaient l'habitude de ne débuter l'analyse et la programmation qu'une fois l'architecture complétée prendront conscience des possibilités et avantages d'une architecture incrémentale. Cela aura des répercussions sur la façon de concevoir l'architecture (Figure 4).

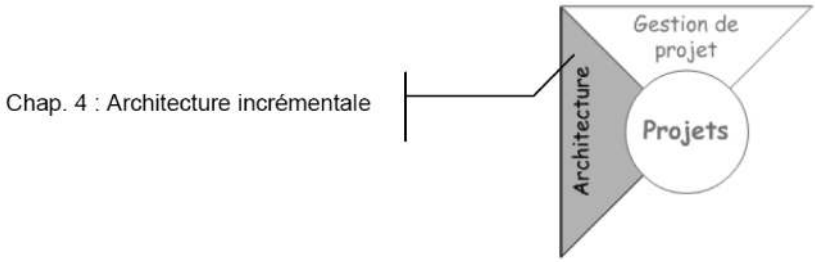


Figure 4 — L'aspect de l'architecture.

Un projet agile est censé livrer le plus tôt possible les fonctionnalités ayant le plus de valeur pour le client. Bien que tous les contextes ne s'y prêtent pas, il est possible et souhaitable de déployer plus souvent des versions incrémentales des applications. Cela a nécessairement un impact sur les utilisateurs qui doivent être formés et informés sur les derniers ajouts et modifications de leur application. Ils seront fort probablement sollicités afin de contribuer à augmenter davantage la valeur des fonctionnalités. Nous avons nommé cet aspect le déploiement (Figure 5).

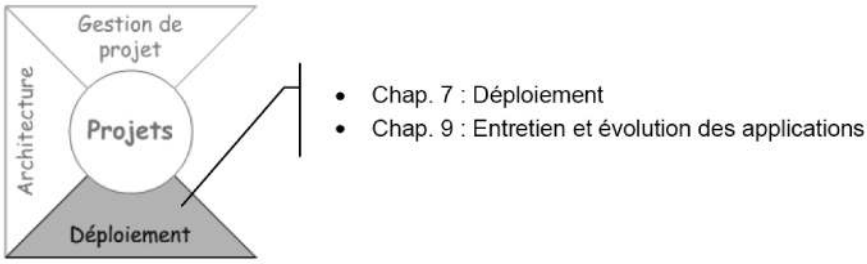


Figure 5 — L'aspect du déploiement.

Tous les projets de développement ou d'évolution de logiciel requièrent une infrastructure technologique, plus ou moins complexe en fonction de la taille de l'organisation utilisatrice, plus ou moins sécurisée en fonction de la sensibilité des informations, et plus ou moins efficace en fonction de la fréquence des déploiements et des changements à apporter. D'abord il y a l'infrastructure de développement, celle utilisée par les équipes de développement et d'évolution. Elle comprend les environnements de tests unitaires, d'intégrations et de vérification. Au final, il y a l'environnement de production ou d'utilisation. Mais entre les deux, il est souhaitable de prévoir un environnement de pré-production. Cet environnement intermédiaire simule l'environnement de production, afin d'y faire des validations. Ces validations sont exécutées par un groupe cible d'utilisateurs, qui testent les

procédures de déploiement. Toute cette infrastructure doit être gérée rigoureusement sous configuration, c'est-à-dire qu'on doit savoir en tout temps quelles sont les versions des applications et autres composantes logicielles qui y sont déployées. Les personnes responsables de l'infrastructure technologique sont immanquablement affectées par la mise en œuvre de l'agilité (Figure 6).

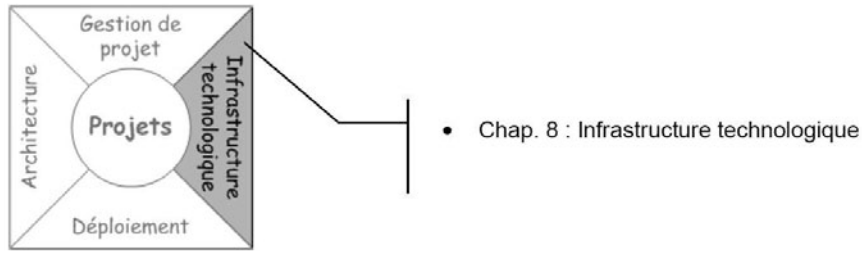


Figure 6 — L'aspect de l'infrastructure technologique.

Processus de troisième niveau : les processus et les individus

Évolution du processus

Plusieurs personnes croient qu'un « processus mature de développement logiciel » et « l'agilité » ne sont pas compatibles. Nous croyons que c'est faux et qu'il est nécessaire d'encadrer les processus de développement logiciel, qu'ils soient agiles ou pas. La canalisation nécessaire à cette évolution est traitée au chapitre *12 Évolution du processus*.

Soutien au développement

À partir d'un certain point, il est normal de vouloir offrir du soutien aux équipes démarrant un projet agile, sous forme d'accompagnement. Cet accompagnement vise à alimenter le cycle d'amélioration continue et à communiquer aux équipes les améliorations apportées. Ce volet est abordé au chapitre *13 Soutien au développement*.

Agilité et documentation

Tous les projets agiles doivent documenter le minimum utile et nécessaire. Ce volet est traité au chapitre *14 Agilité et documentation*.

Nous avons regroupé ces pratiques sous l'aspect du soutien au développement (Figure 7).

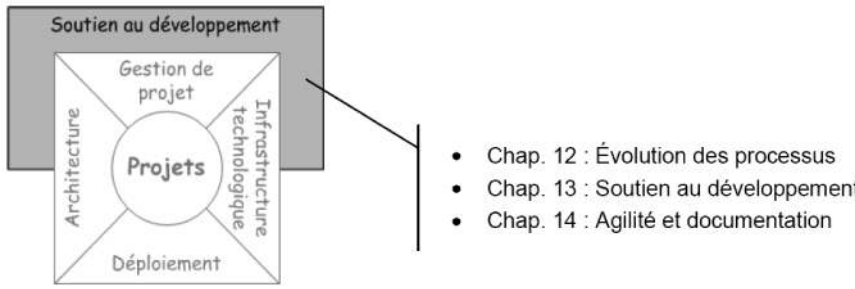


Figure 7 — L'aspect du soutien au développement.

Gestion des individus

À partir du moment où une organisation se transforme en adoptant les valeurs et les principes agiles, elle peut être appelée à transformer aussi sa gestion des individus (Figure 8), soit la dotation de personnel, l'évaluation des compétences et des performances, les cheminements de carrière, le remerciement de personnel, et les attributions de salaire et bonus. De plus, il est important de considérer le savoir-être des individus en place qui pourrait dominer sur leur savoir-faire, et ce pour la majorité des points couverts par la gestion des individus, tant pour le succès des projets que pour développer des cohésions d'équipe.

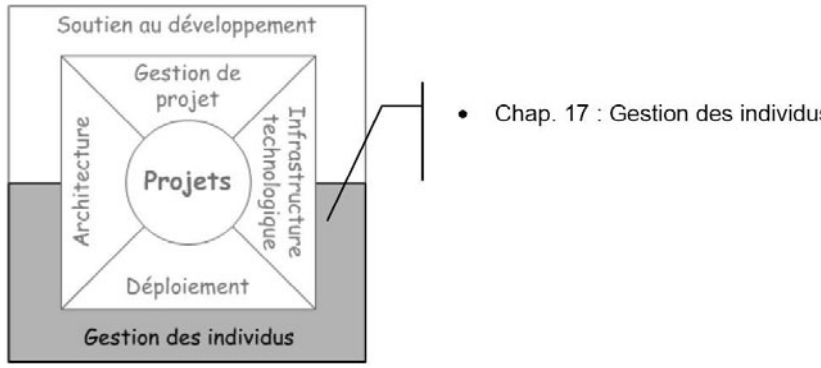


Figure 8 — L'aspect de la gestion des individus et des compétences.

Processus de quatrième niveau : la gouvernance

Les dirigeants sont responsables de la performance de leur organisation. Ils en définissent la structure et utilisent la reddition de compte pour alimenter leur processus de décision. Ils apportent à leurs équipes tout ce dont elles ont besoin pour mener à bien la mission de l'organisation :

- vision ;
- objectifs ;

- motivation ;
- ressources humaines et monétaires ;
- plans à haut niveau ;
- etc.

Ces dirigeants sont souvent responsables de la gestion du portefeuille applicatif, tant pour leur adéquation aux besoins d'affaires que pour la gestion du coût d'entretien et d'évolution, en passant par la pérennité des technologies utilisées et l'efficacité du processus de développement.

Ils doivent également mettre en place une saine atmosphère de travail qui favorisera le travail d'équipe et le développement du potentiel des individus. Nous avons regroupé ces responsabilités sous l'aspect de la « gouvernance » (Figure 9).

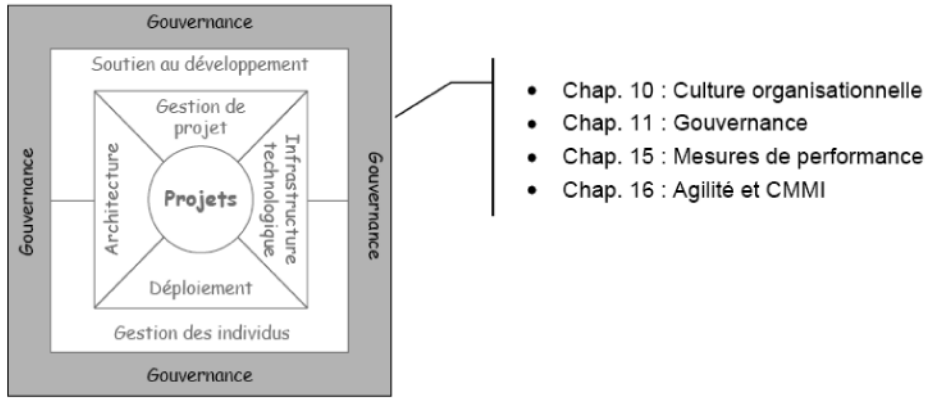


Figure 9 — L'aspect de la gouvernance qui chapeaute le processus organisationnel.

LES MISES EN SITUATION ET LES RETOURS D'EXPÉRIENCE

Ce livre comprend plusieurs « mises en situation » et « retours d'expérience », disséminés dans chacun des chapitres, racontant des histoires de personnes ayant fait face à des situations particulières liées à leur choix de l'agilité. Toutes les mises en situation présentées dans cet ouvrage sont inspirées de cas réels rencontrés dans notre pratique professionnelle. Tous les retours d'expérience sont des cas réels vécus par l'un de nous, parfois les deux. Nous avons souhaité partager avec vous ces mini-études de cas, pas seulement pour les descriptions des problématiques ou des opportunités, mais aussi pour les solutions apportées.

Bien que les domaines d'affaires des organisations soient mentionnés, les noms des organisations ont été omis et les prénoms des personnes ont été changés pour conserver le caractère confidentiel de chacun. Nous vous présentons ici la première mise en situation.

Mise en situation : Charles et la transition à l'agilité

Charles est le chef de la section Technologies de l'Information (TI) d'une organisation gouvernementale. La plupart des systèmes informatiques supportant les opérations ont été conçues voilà plus de trente ans. Les systèmes plus récents ont tout de même plus de dix ans mais leur technologie n'est plus supportée. Une réingénierie s'impose.

L'équipe de Charles a travaillé pendant plus de six mois à l'analyse des nouveaux systèmes sans pouvoir livrer quoi que soit. La complexité grandissante qu'elle a découverte a fait en sorte qu'elle n'est pas arrivée à compléter l'analyse détaillée[1]. Le cycle de vie en cascades n'a pas aidé : lourdeur dans la documentation, carcan dans les rôles et responsabilités liés aux titres des personnes, difficulté à établir des preuves de concept et manque de réactivité face aux changements du processus métier des utilisateurs. Charles a tôt fait de réaliser qu'il faut procéder autrement pour gérer la complexité et arriver à livrer des petits morceaux, plus souvent, et ainsi tabler sur des succès.

Hormis la réingénierie nécessaire des applications, Charles doit aussi procéder à la réingénierie des processus de développement. Il sait qu'il devra réunir les efforts de ses équipes TI avec ceux des équipes métiers, se rapprocher de ses clients et utilisateurs afin de produire des applications qui comblent leurs besoins qui ont évolué au cours des dernières années.

Charles a alors entendu parler des méthodes agiles et de leur philosophie. Il a été surtout séduit par les résultats obtenus par les organisations les ayant mises en place. Il sait cependant que d'implanter de telles méthodes dans une organisation gouvernementale est en soi un défi de taille qui n'a pas encore été tenté à large échelle.

Charles va rapidement constater qu'un changement fondamental de la philosophie de travail aura des répercussions dans toutes les sphères de sa direction générale :

– d'abord les projets eux-mêmes, mais aussi la gestion de projets et la reddition de comptes qui en émane ;
– la façon de faire l'architecture ;
– les façons de se coordonner avec l'équipe d'infrastructure technologique ;
– la gouvernance et l'élaboration de stratégies combinées affaires-TI ;
– la gestion des processus de développement et d'évolution ;
– la gestion de la transition et du changement ;
– la gestion des individus et des compétences.

Charles est confiant que cette transition sera facilitée par l'affectation d'accompagnateurs experts en agilité. Et c'est à leur arrivée que s'amorce cette transition.

1. C'est ce qu'on appelle un cas classique de « paralysie d'analyse »– traduction libre de l'expression américaine « *analysis paralysis* » –où l'équipe s'embourbe et s'enlise dans l'analyse sans espoir d'en sortir en moins d'une année.

NOTRE INTENTION AVEC CET OUVRAGE

Avec ce livre, nous aimerions partager avec vous notre expérience et vous aider à modifier, ou créer un processus de développement qui convienne à votre organisation et qui profite des principes et valeurs des méthodes agiles. Nous espérons que cet ouvrage vous sera utile pour réfléchir au contexte de votre organisation et trouver la meilleure façon d'adopter une approche agile.

REMERCIEMENTS

Le soutien de nombreuses personnes est absolument requis pour réussir à publier un tel ouvrage. C'est pourquoi nous tenons à les remercier sincèrement d'avoir permis de mener notre projet à terme.

Nos familles

L'écriture de cet ouvrage a requis de nombreuses heures volées à nos familles respectives et les remercions pour leur soutien indéfectible : Véronique Lambert, Héloïse, Pénélope, et Magalie Boisvert ainsi qu'Yves Duchemin.

Nos muses

Plusieurs personnes nous ont largement inspirés et nous les remercions, certains pour la générosité dans le partage de leurs connaissances et compétences, d'autres pour nous avoir légué des ouvrages extraordinaires. Il y a eu nos collègues de Pyxis Technologies. D'abord François Beauregard, le fondateur, pour avoir inspiré le chapitre sur la culture organisationnelle avec sa présentation « *De l'agilité qui réduit l'agilité* » lors de l'Agile Tour 2010 à Montréal. Puis il y a eu Yves Ferland, leader du Pyxis Coach Team, Martin Proulx, le président, et Isabelle Therrien, coach agile. Les discussions que nous avons eues ont alimenté plusieurs de nos chapitres.

Ensuite, viennent quelques collègues de l'industrie, dont Alain Abran, professeur à l'École de Technologie Supérieure et directeur de thèse de doctorat de Sylvie, Michel Martel, président d'Analystik, et Danielle Ouellet de Production DO3. Vos compétences, appliquées au quotidien, sont une source d'inspiration inouïe.

Finalement, il y a eu nos auteurs préférés, Mike Cohn, Amr Elssamadisy, Jeff Paton, et Scott Ambler dont les ouvrages sont cités dans nos *références bibliographiques*.

Nos relecteurs

Pour assurer la pertinence et la justesse du contenu des chapitres, nous avons fait appel à nos collègues de l'industrie qui ont gracieusement accepté de réviser les chapitres. Sans leur apport, cet ouvrage n'aurait pas pu être ce que nous en avons fait. Nous les savions tous très occupés et les remercions d'autant plus pour leur contribution bénévole.

Tout d'abord, il y a eu nos collègues de Pyxis Technologies. Elsa Larreur et Tremeur Balbous, nos Bretons favoris et tous deux coaches agiles, ont contribué à améliorer la qualité de notre français écrit en plus d'apporter des commentaires toujours pertinents. Isabelle Therrien, coach agile, qu'aurions-nous fait sans toi ? Tu as révisé cinq chapitres, et pas les plus petits ! Tes commentaires se sont comptés par centaines et la qualité de tes révisions est indéniable. François Beauregard, fondateur de Pyxis et coach agile et organisationnel, Martin Proulx, président de Pyxis, vos révisions nous ont fait progresser tant du côté du contenu que du côté personnel. Vincent Cléroux et Vincent Tencé, coaches agiles, merci d'avoir volontairement proposé votre aide pour les révisions car votre implication tombait à un moment où nous en avions le plus besoin. Éric Laramée, coach agile et Scrum Master, merci pour ta contribution qui fut très appréciée. Tu as été le déclencheur de cet ouvrage.

Ensuite, il y a des collègues de l'industrie que nous avons eu le bonheur de côtoyer au cours de nos diverses missions. Bruno Bouchard, travaillant présentement pour TD au sein du groupe de méthodologie de livraison entreprise TDI-ST, tes révisions ont été toutes aussi pertinentes que les propos que nous avons le plaisir d'échanger lors de nos rencontres aux présentations d'Agile Montréal. Michel Bergeron, conseiller en gestion du changement et de la formation, merci pour le partage de ton expertise. Francine Lauzon et Danielle Ouellet, toutes deux gestionnaires hors pair avec lesquelles nous avons beaucoup de plaisir à travailler.

Finalement, un grand merci à Alain Abran et Alain April, tous deux auteurs de livres et de nombreuses publications et professeurs de génie logiciel à l'École de Technologie Supérieure. Votre feedback a été des plus utiles. Nous avons particulièrement apprécié votre générosité et la vitesse avec laquelle vous nous retourniez vos commentaires.

Notre éditeur

Finalement, nous tenons à remercier notre éditeur, Jean-Luc Blanc, pour sa patience et sa compréhension au cours de notre première expérience d'auteurs de livre. Ses conseils nous ont permis de publier un meilleur travail.

1

Qu'est-ce que l'agilité ?

Objectif

Dans ce chapitre, vous découvrirez pourquoi l'adoption de méthodes agiles au sein d'une organisation qui développe et entretient des logiciels requiert un encadrement et une transformation des processus organisationnels. Une transition est requise et elle se doit d'être planifiée, communiquée, évolutive et suivie, afin de bénéficier des avantages qu'apportent les méthodes agiles.

Mise en situation : Jacques et son questionnement sur l'agilité

Autour d'une machine à café, Jacques, un coordonnateur d'équipes de développement et Stéphane, un coach agile, discutent.

Jacques : *Peut-on choisir « l'agilité » et n'impacter que les équipes de développement ?*

Stéphane : *Cela ne serait pas « agile ».*

Jacques : *Peut-être, mais si on commençait par les équipes de développement, cela devrait fonctionner et nous pourrions prétendre « être agile »...*

Stéphane : *C'est un bon point de départ, mais insuffisant pour être soutenu à moyen et long termes, et pour apporter les bénéfices et avantages de l'agilité. Alors revenons à la base : quelles sont les raisons qui vous ont mené à choisir l'agilité ?*

Jacques : *Il y en a plusieurs : motivation des équipes en déclin, dépassements budgétaires et des échéanciers, difficultés à mesurer l'avancement réel des projets, fonctionnalités livrées qui ne répondent pas aux besoins, coûts de plus en plus élevés pour effectuer des changements même mineurs, difficulté à recruter du personnel compétent et motivé. Bien, c'est tout un tas de raisons.*

Stéphane : *Je comprends que vous venez d'énoncer une liste de symptômes. En effet, il est plus probable que ces symptômes diminuent en mettant en place l'agilité qu'en conservant le statu quo. Mais les vraies raisons, quelles sont-elles ?*

Jacques : *Nous nous sommes rendu compte que nos façons de faire, nos processus, avaient pris de la lourdeur avec le temps, à un point où nous ne faisons plus les choses efficacement. De plus, nous subissons la pression des cadres supérieurs pour passer à l'agilité dès que possible.*

Stéphane : *On ne choisit pas l'agilité pour faire de l'agilité. On choisit l'agilité parce qu'on croit que les principes et les valeurs qu'elle apporte vont nous permettre de livrer fréquemment et efficacement des solutions logicielles qui répondent aux besoins, de façon à rentabiliser les investissements du client le plus tôt possible.*

Jacques : *Mais qu'est-ce qu'implique le choix de l'agilité ?*

Stéphane : *Choisir l'agilité implique une période de transition qui affectera une partie importante de l'organisation. Cela implique qu'une partie des processus actuels va subir des transformations, que les rôles des individus vont aussi être transformés.*

1.1 LE CHOIX DE L'AGILITÉ

Votre organisation songe à adopter l'agilité – sa philosophie, ses principes, ses valeurs, quelques-unes de ses méthodes – pour une raison qui lui est propre, que cette raison soit précise ou encore floue. Vous avez entendu parler de *Scrum*, de *Extreme Programming* (XP), de *Test Driven Development* (TDD), quelques-unes de ces méthodes agiles émergeant au début des années 2000 et qui sont devenues des incontournables une décennie plus tard.

Même si vous avez déjà eu un aperçu de ce qu'est l'agilité ou de l'une des méthodes la mettant en œuvre, il peut être utile de faire un bref retour sur cette notion d'agilité au-delà des notions d'une méthode en particulier.

L'agilité est une philosophie de travail basée sur des valeurs et des principes. Les méthodes et approches qui sont dites « agiles » sont cohérentes avec ces valeurs et ces principes.

1.2 LES QUATRE VALEURS DE L'AGILITÉ

Pour être considérée agile, une approche doit répondre au *Manifeste agile*, qui se découpe en quatre valeurs et en douze principes. L'agilité valorise[1] :

1. **Les individus et les interactions davantage que les processus et les outils** : l'hypothèse de cette valeur est que le succès d'un développement logiciel est beaucoup plus dépendant des individus y travaillant que des techniques et des technologies utilisées, même si les processus et les outils peuvent contribuer à l'efficacité du travail.

1. Tirées du manifeste agile sur Wikipedia, http://fr.wikipedia.org/wiki/Manifeste_agile.

2. **Les logiciels fonctionnels davantage qu'une documentation exhaustive** : l'hypothèse de cette valeur est qu'il n'existe pas de meilleure façon de mesurer l'état d'avancement d'un projet que de vérifier à intervalles réguliers ce que le système est déjà capable de faire, en comparaison avec l'état d'avancement d'une liste de tâches décomposées et illustrées, par exemple, sur un graphique Gantt.
3. **La collaboration avec le client davantage que la négociation de contrat** : l'hypothèse de cette valeur est qu'il est préférable de travailler sur la relation et la collaboration entre le fournisseur et le client, plutôt que de construire un contrat strict qui servira de preuve pour prendre en défaut l'une ou l'autre des parties.
4. **La réponse au changement davantage que le suivi d'un plan** : l'hypothèse de cette valeur est qu'il est préférable de produire le meilleur système, quitte à développer un système différent que celui qu'on a pu imaginer en définissant un plan de livraison au commencement du projet.

Nous avons décrit les valeurs sous forme d'hypothèses, parce qu'il est possible d'appliquer les pratiques des approches agiles sans au préalable avoir pris comme hypothèse les valeurs de l'agilité. Si votre organisation décide de prendre le virage agile sans prendre comme hypothèse les valeurs du manifeste, vous ouvrez la porte à l'argumentation de la part des gens qui ne comprendront pas les raisons premières des approches agiles.

1.3 LES DOUZE PRINCIPES DE L'AGILITÉ

Des valeurs du manifeste découlent ces douze principes [1] :

1. **Notre première priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels utiles** : ce principe est derrière le caractère incrémental des approches agiles, qui veut qu'à la fin de chaque itération, le client puisse vérifier une nouvelle version fonctionnelle et améliorée du système en cours de développement.
2. **Le changement est accepté, même tardivement dans le développement. Les processus agiles exploitent le changement comme avantage compétitif pour le client** : ce principe est derrière les carnets de produit, qui permettent au client de planifier lui-même la réalisation de son système, en fonction de l'état d'avancement de son projet, du résultat obtenu à la dernière itération, des opportunités et des imprévus du moment présent.
3. **Livrer fréquemment une application fonctionnelle, toutes les deux semaines à deux mois, avec une tendance pour la période la plus courte** : ce principe est derrière le concept des itérations, soit des périodes de temps fixes servant de

1. Tirés tel quel du manifeste agile sur Wikipedia, http://fr.wikipedia.org/wiki/Manifeste_agile, consulté en octobre 2010. Nous expliquons les hypothèses derrière chacun.

points de visibilité et permettant de planifier le projet, de valider les résultats obtenus, et d'améliorer les façons de faire.

4. **Les experts métier et les développeurs doivent collaborer quotidiennement au projet** : ce principe explique que le client est un membre à part entière des équipes des projets, parce qu'il est le meilleur représentant pour planifier le projet, recueillir les besoins et décrire les fonctionnalités.
5. **Bâtissez le projet autour de personnes motivées. Donnez-leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail** : ce principe explique le changement dans le mode d'intervention envers les équipes de projets. Un mode d'intervention qui s'assure de réunir les conditions gagnantes au moment de constituer une équipe, qui fait confiance aux équipes en évitant l'ingérence, qui demande de la transparence de la part des équipes tout en donnant le droit à l'erreur, et qui apporte un appui lorsque les équipes rencontrent des difficultés hors de leur contrôle.
6. **La méthode la plus efficace pour transmettre l'information est une conversation en face à face** : ce principe explique que les documents sont de bons moyens d'assurer la pérennité et la conformité d'un système, mais qu'ils sont un moyen pauvre de communication.
7. **Un logiciel fonctionnel est la meilleure unité de mesure de la progression du projet** : comme expliqué précédemment, ce principe souligne qu'il est plus utile de mesurer l'état d'avancement d'un projet par le nombre de fonctionnalités livrées, que par le suivi des activités sur un graphique de Gantt ou le suivi seul du pourcentage de consommation des coûts de projet sans le comparer à ce qui est livré et fonctionnel.
8. **Les processus agiles promeuvent un rythme de développement durable. Commanditaires, développeurs et utilisateurs devraient pouvoir maintenir le rythme indéfiniment** : un principe qui demande un rythme soutenu et soutenable, qui n'épuise pas les équipes et qui leur donne le temps de développer des solutions de qualité de façon prévisible.
9. **Une attention continue à l'excellence technique et à la qualité de la conception améliore l'agilité** : un principe qui considère que les échéanciers, les ressources budgétaires, la portée d'un projet sont des leviers pour ajuster le déroulement d'un projet, mais pas la qualité. Une qualité escamotée pour atteindre d'autres seuils dissimule du travail restant et gonfle artificiellement la capacité réelle des équipes de projets.
10. **La simplicité – l'art de maximiser la quantité de travail à ne pas faire – est essentielle**[1] : ce principe encourage à se limiter à l'essentiel, à prendre les décisions au « dernier moment responsable »[2]pour éviter d'investir dans

1. Plusieurs personnes ont suggéré de remplacer l'énoncé de ce principe par « l'art d'éviter le travail inutile ». Nous avons sciemment laissé ce principe tel quel.

2. Le « dernier moment responsable » est le moment le plus tard possible sans conséquences négatives.

des efforts qui, au final, ne seront pas utiles ou seront considérés comme du gaspillage de temps et d'argent.

11. **Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto-organisent** : ce principe encourage l'intelligence collective et évite que les spécialistes, comme les architectes, deviennent des goulots d'étranglement aux engagements d'équipes et nuisent à leur autonomie.
12. **À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accorde et ajuste son comportement dans ce sens** : ce principe est celui de l'amélioration continue. Pourquoi faire des post-mortem à la fin des projets pour servir de recommandations aux prochaines équipes ? Il est plus utile de s'améliorer sur une base continuelle, d'utiliser les itérations comme des points de visibilité et de s'améliorer tout au long de son propre projet.

Toutes les méthodes agiles ont en commun les valeurs et principes[1] du manifeste. Vous pouvez vous inspirer d'une ou plusieurs approches agiles pour construire votre propre processus organisationnel. Mais pensez à contre-vérifier vos décisions avec le manifeste lorsque vous voulez modifier ou abandonner une pratique agile. Peut-être êtes-vous en train de cacher un problème rendu visible.

Pour une description des différentes méthodes agiles et pour expliquer ce qui les distingue l'une de l'autre, vous pouvez consulter le chapitre *2 Les méthodes agiles*.

1.4 AVANTAGES ET BIENFAITS

Il existe plusieurs raisons pour lesquelles les organisations adoptent l'agilité. L'historique des échecs et des défis de leurs précédents projets de développement les ont incitées à changer de façon de faire. Les processus qui avaient été appliqués n'ont pas donné le succès escompté. Dans la conversation en début de chapitre, nous avons fait état des symptômes vécus par certains projets et de quelques raisons qui poussent les organisations à adopter l'agilité.

1.4.1 Avantages

Lors de la conférence XP 2002, Jim Johnson du Standish Group, a présenté des statistiques troublantes sur l'utilisation des fonctionnalités développées dans l'industrie du logiciel (voir Figure 1.1).

1. Lorsqu'une qu'une section du livre se rapporte directement à un principe ou une valeur agile, nous allons la mettre en lumière dans le texte pour souligner son application.

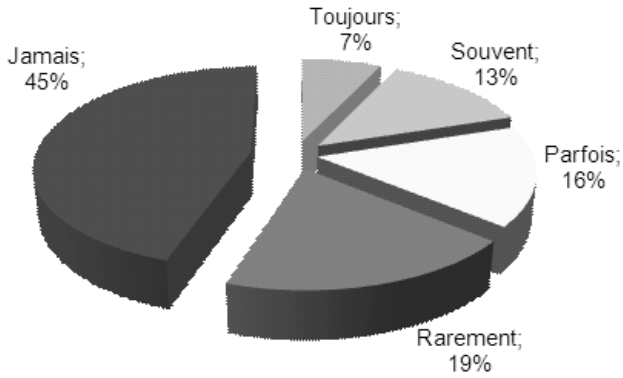


Figure 1.1 — Statistiques d'utilisation des fonctionnalités.

Ce qui est troublant, ce sont les 45 % de fonctionnalités développées mais jamais utilisées. Le développement de ces fonctionnalités a consommé une quantité considérable de ressources financières et de compétences humaines pour n'être **jamais utilisées.** De ce constat généralisé de l'industrie découle le premier principe agile et, par conséquent, le premier avantage à « agiliser » ses projets, soit de livrer de la valeur d'affaires au client le plus tôt possible.

La liste ci-dessous résume les avantages pour le client découlant de l'adoption de l'agilité par ses équipes de développement.

- **Livrer de la valeur d'affaires au client** : en développant des solutions utiles pour les utilisateurs.
- **Améliorer la qualité** : en réduisant l'inventaire des anomalies et leur coût inhérent de gestion.
- **Réduire les délais de mise en production** : en livrant fréquemment des incréments de logiciel fonctionnel, en ordre décroissant de valeur d'affaire.
- **Respecter le budget et les dates de livraison** : en réduisant l'incertitude (les risques) le plus tôt possible et en surveillant sa vélocité, l'équipe peut, en toute transparence, livrer les fonctionnalités à l'intérieur de l'enveloppe budgétaire et des dates de livraison qu'elle aura participé à établir avec le client.
- **Améliorer la collaboration et la communication** : en incluant un représentant du client dans l'équipe, en mettant en place une équipe auto-organisée et assurant la transparence.

1.4.2 Bienfaits

Certains bienfaits pourront être observés en tant que conséquences aux avantages, soit :

- **Amélioration de la productivité** : nous avons mesuré la productivité de plusieurs dizaines de projets de développement logiciel dans un certain nombre d'organisations, dont plusieurs faisaient une transition à l'agilité. Pour en savoir

plus sur l'amélioration de la productivité, nous vous invitons à consulter le chapitre *15 Mesures de performance*.

- **Réduction des coûts** : Une planification empirique priorisant le développement par ordre de valeur d'affaires réduit les coûts de développement. Cela s'explique par une économie dans le développement de fonctionnalités peu ou pas utiles. Donc, la situation où 45 % de fonctionnalités livrées ne sont jamais utilisées devient peu probable.
- **Périodes de stabilisation plus courtes** : en améliorant la qualité des fonctionnalités livrées à chacune des itérations et en adoptant l'intégration continue, les équipes voient la période de stabilisation précédant une mise en production passer de plusieurs semaines, voire plusieurs mois, à quelques jours et même quelques heures dans certains cas.
- **Meilleure motivation des membres d'équipe** : en livrant fréquemment du logiciel fonctionnel et de qualité et en respectant ses engagements, les membres d'équipe gagnent confiance en leurs capacités et développent des sentiments d'accomplissement et de fierté. Cela augmente naturellement leur motivation. De plus, nous savons tous qu'un employé motivé est un employé productif.

En résumé

L'adoption de méthodes agiles dans certains projets de développement logiciel ne suffit pas pour tirer le maximum d'avantages qu'apporte l'agilité à l'échelle organisationnelle. Une organisation adoptant l'agilité n'a pas non plus l'obligation de faire tous ses projets en mode agile. Mais pour les projets qu'elle décidera de faire en mode agile, il faudra bien comprendre les impacts de l'adoption de l'agilité. Une phase de transition s'avère nécessaire, car il s'agit d'une gestion normale de changement, pour permettre des ajustements dans tous les aspects de l'organisation potentiellement impactés par l'agilité

2

Les méthodes agiles

Objectif

Dans ce chapitre, vous découvrirez les spécificités et les pratiques communes de chacune des méthodes agiles. Les éléments à considérer au moment de sélectionner un projet candidat afin d'optimiser l'introduction de l'agilité dans votre organisation sont étudiés en fin de chapitre.

Mise en situation : Christophe et son choix inadéquat de méthode agile

Christophe est un conseiller expert en agilité. Il maîtrise les méthodes Scrum et *eXtreme Programming*, car il a tenu tour à tour les rôles de Scrum Master et de développeur dans des équipes de développement agile.

Aujourd'hui, Christophe fait face à un nouveau défi : les méthodes qu'il maîtrise ne semblent pas convenir à son principal client, une grande entreprise bancaire. La direction de l'entreprise a une réelle volonté à devenir agile, mais elle redoute que cela implique de renoncer complètement à son processus organisationnel. La direction reconnaît qu'elle doit évoluer pour s'améliorer, mais refuse de se soustraire de tous ses rôles, processus et livrables. Le processus de l'organisation est basé sur la méthode *Rational Unified Process* (RUP), qui couvre de manière détaillée tout le cycle des projets de développement et concerne toutes les disciplines du développement logiciel.

Christophe trouve difficile de rassurer les membres de l'organisation, qui ne trouvent aucun point de repère dans les méthodes Scrum et XP. Cherchant une solution à cette difficulté, il découvre une version agile de la méthode RUP : *Agile UP*. Après vérification, il s'aperçoit que cette méthode provient de l'un des mentors de la communauté agile et qu'elle répond correctement à la philosophie de l'agilité.

Christophe décide donc de revoir son approche et il prépare une présentation à l'intention de la direction afin de leur partager sa découverte.

2.1 LES MÉTHODES AGILES ET LEURS PARTICULARITÉS

La littérature traitant des méthodes agiles se concentre sur le développement logiciel et en particulier sur les équipes et les projets. Alors que cela peut être frustrant pour les gestionnaires cherchant à comprendre les répercussions de cette transition sur leur organisation, la compréhension des pratiques et méthodes agiles est le premier pas pour entamer une transition organisationnelle.

Le présent chapitre décrit les généralités des méthodes agiles qui constitueront le noyau des pratiques proposées dans ce livre[1]. À partir de ce noyau de pratiques, vous serez en mesure d'analyser les impacts d'adopter l'agilité sur la globalité de votre organisation : le bureau de projet, l'équipe d'architecture, le comité de produit, etc. La suite du livre vous aidera à compléter cette analyse qui dépasse les seuls impacts sur les équipes de développement.

Toutes les méthodes agiles reposent sur des valeurs exprimées dans le *Manifeste agile*. Elles se concentrent cependant sur des aspects différents des problématiques logicielles.

2.1.1 Scrum

La méthode la plus adoptée aujourd'hui pour faire de la gestion de projet et de la gestion d'exigences. Son premier objectif est d'augmenter la qualité de la communication et de développer des solutions utiles :

- en encourageant les équipes auto gérées et multidisciplinaires ;
- en rapprochant le client de l'équipe de développement ;
- en planifiant la réalisation des exigences par ordre de valeur d'affaires.

Elle est peu restrictive et demande de la rigueur pour être bien appliquée.

C'est une méthode à considérer pour augmenter l'engagement des équipes de développement, accroître la collaboration avec les clients et développer le système avec la plus grande valeur ajoutée possible au lieu de répondre intégralement à un cahier de spécifications qui pourrait ne pas convenir pleinement aux besoins.

2.1.2 XP (eXtreme Programming)

La méthode agile la plus technique. Elle encourage de rigoureuses pratiques d'ingénierie logicielle qui augmentent la qualité et la souplesse du code développé :

- en proposant les meilleures pratiques reconnues d'ingénierie logicielle (développement piloté par les tests, intégration continue, restructuration de code [*refactoring*], etc.) ;

1. Pour une description détaillée des méthodes et pratiques, nous vous invitons à compléter votre lecture avec les ouvrages que nous avons regroupé dans la bibliographie, en particulier la section *Agilité et pratiques agiles*.

- en proposant des pratiques de collaboration entre les membres de l'équipe de programmation (notamment par du binômage[1] (*pair programming*) et la propriété collective du code).

C'est une méthode à considérer pour améliorer la qualité technique d'un développement logiciel :

- en augmentant la confiance des développeurs envers leur code ;
- en augmentant le niveau d'assurance des développeurs à restructurer le code sans risquer d'introduire des régressions (défauts) dans les produits ;
- en s'imposant d'écrire du code clair et propre permettant d'avoir une vitesse de développement soutenue et réaliste.

2.1.3 Lean/Kanban

Une méthode directement inspirée des méthodes manufacturières ayant fait le succès de compagnies tel que Toyota. Elle applique l'amélioration continue de l'efficacité des processus de développement et vise l'élimination du gaspillage :

- en appliquant l'amélioration continue pour augmenter l'efficacité des processus de développement ;
- en livrant peu, mais fréquemment, de manière à réduire l'inventaire du développement logiciel ;
- en repoussant les décisions au dernier moment responsable, soit le plus tard possible sans conséquence sur les résultats ;
- en diminuant le gaspillage des efforts et les temps d'attentes des processus.

C'est une méthode à considérer lorsque le retour sur investissement des processus de développement n'est pas jugé satisfaisant. Elle facilite également l'adoption d'une approche agile, lorsque les experts spécialisés de votre organisation sont un frein à la constitution d'équipes pluridisciplinaires par leur manque de disponibilité.

2.1.4 Agile UP

Une adaptation de la méthode RUP (*Rational Unified Process*) à l'agilité. Elle est la plus prescriptive des méthodes agiles et elle convient probablement mieux aux grandes organisations qu'aux petites. Elle propose quatre grandes périodes itératives et incrémentales dans le cycle de développement logiciel :

- **Évaluation** *(inception)* : pour définir les objectifs et le périmètre du projet ;
- **Élaboration** *(elaboration)* : pour les preuves de concepts et la construction initiale de l'architecture ;
- **Construction** : pour la réalisation du projet ;
- **Transition** : pour la livraison du produit.

1. Le binômage est décrit plus loin dans la partie « Pratiques de développement » de la section 2.2.3.

C'est une option à considérer lorsque la culture organisationnelle nécessite une méthode plus structurante que les méthodes agiles moins prescriptives.

2.1.5 Crystal

Crystal regroupe une famille de méthodes agiles adaptées à la taille et à la criticité des projets, classifiées par couleur :

- *Sapphire* et *Diamond* pour les grands projets à haut risque ;
- *Clear*[1], *Yellow* et *Orange* pour les projets de plus petite envergure.

L'idée derrière le choix d'une méthode *Crystal* est qu'il n'existe pas une seule méthode convenant à tous les contextes de projets. Des projets de cent personnes distribuées en plusieurs équipes, dans plusieurs pays, n'ont pas les mêmes enjeux qu'un projet de dix personnes localisées dans le même édifice. Les projets ayant un impact sur la vie humaine n'ont pas les mêmes enjeux que les logiciels de divertissement.

Ce sont des méthodes à considérer lorsque l'organisation comporte plusieurs contextes différents de développement et qu'il est souhaité d'adopter une approche spécifique à chacun de ces contextes.

2.2 LES PRATIQUES ET LIVRABLES AGILES

Certaines pratiques et livrables sont plus populaires puisqu'elles sont présentes dans plusieurs méthodes agiles. Voici quelques pratiques auxquelles une organisation sera probablement exposée, indépendamment des méthodes agiles qu'elle a choisi d'adopter.

2.2.1 Sprint 0 (démarrage de projet)

À part la méthode Agile UP, les méthodes agiles sont très peu prescriptives sur les pratiques à adopter pour démarrer un projet. Il existe de la documentation sur l'itération 0 (*Sprint zéro*) sur Internet, mais aucune méthode ne la reconnaît officiellement[2].

2.2.2 Pratiques de gestion

Les pratiques agiles proposent des techniques de gestion différentes des méthodes traditionnelles. Ces pratiques sont adaptées à la gestion incrémentale dont l'objectif principal est d'utiliser la solution logicielle fonctionnelle comme unité de suivi des

1. Alistair Cockburn, *Crystal Clear: A Human-Powered Methodology for Small Teams*, Addison-Wesley, 2004.
2. Pour plus de détails sur la façon de personnaliser le démarrage de projet, nous vous invitons à consulter le chapitre 3 *Démarrage d'un projet agile*.

projets de développement. Voici quelques définitions de pratiques habituellement présentes en gestion de projet agile[1] :

- les **itérations** (connues aussi sous le nom de *cycles* ou *sprints*) : des périodes courtes de quelques semaines pendant lesquelles une équipe réalise un nouvel incrément du produit logiciel. La portée de cet incrément provient de la séance de planification au tout début de l'itération ;
- les **séances de planification** : la séance pendant laquelle l'équipe de projet s'engage auprès du client sur la quantité et la nature des fonctionnalités à être développées et livrées au terme de l'itération ;
- les **mêlées quotidiennes** (connues aussi sous le nom de *points quotidiens de synchronisation*, *Scrum meetings* ou *Daily stand-up*) : rencontres quotidiennes de quelques minutes pendant lesquelles l'équipe se synchronise, planifie la prochaine journée et identifie les obstacles de l'itération en cours ;
- les **revues d'itération** : rencontres de synchronisation de fin d'itération pendant lesquelles l'état d'avancement du projet est vérifié. Le dernier incrément du produit est présenté sous la forme d'une démonstration. Les observations de la démonstration, jumelées au carnet de produit, servent d'éléments d'information à la planification de l'itération suivante ;
- les **rétrospectives** : moment où l'équipe s'arrête pour faire une introspection sur ses façons de faire et se doter d'un plan d'action SMART[2] pour s'améliorer ;
- le **Poker planning©**[3] : une technique d'estimation par affinité fournissant une évaluation relative (en *Story points* ou points d'effort) d'un item du carnet de produit, combinant l'effort nécessaire, l'apprentissage potentiel et le niveau de risque associé. L'estimation ne prend pas directement en compte le temps de développement (ex. les jours-personnes).

2.2.3 Pratiques de développement

La livraison régulière d'incréments de logiciel fonctionnel a un impact sur les pratiques de développement. En plus d'effectuer toutes les disciplines du développement à l'intérieur des itérations, les équipes ont besoin de nouvelles pratiques assurant la qualité du produit pendant tout le déroulement du projet. Voici quelques définitions de pratiques de développement logiciel utilisées dans un contexte agile[4]:

- le **TDD** (*Test Driven Development*) : technique de programmation consistant à coder un test vérifiant le comportement attendu d'une méthode du code. Une fois le test écrit, le développeur implémente la méthode dans le code jusqu'à ce que le résultat du test soit positif ;

1. Pour plus de détails sur les pratiques de gestion, nous vous invitons à consulter les chapitres *5 Équipes et livraison incrémentale* et *6 Gestion de projet*.
2. SMART: *Spécifique, Mesurable, Atteignable, peRtinent* (*Relevant*) et limité dans le Temps.
3. © Mountain Goat Software.
4. Pour plus de détails sur les pratiques de développement d'un projet, se référer au chapitre *4 Architecture incrémentale*.

- la **restructuration du code** (*Refactoring*) : approche favorisant l'entretien régulier du code de manière à le garder clair et souple ;
- le **binômage** (*Pair Programming*) : pratique consistant à programmer à deux développeurs sur le même poste de travail. Pendant que l'un code et explique sa démarche, le second fait de la revue de code en direct. Les développeurs intervertissent leurs rôles régulièrement ;
- l'**intégration continue** : pratique consistant à compiler le code, inclure les changements et exécuter le plan de tests automatisés à chacune des intégrations de code. Cela permet de découvrir rapidement et fréquemment l'introduction de régressions et les problèmes potentiels d'intégration. Pour profiter pleinement de ces avantages, les développeurs soumettent leur code le plus fréquemment possible et un serveur d'intégration est configuré pour contrôler automatiquement l'intégration de chacune de ces soumissions.

2.2.4 Livrables agiles

La plupart des livrables demeurent les mêmes dans un contexte agile, mis à part quelques nouveaux livrables, mieux adaptés au développement incrémental. Voici quelques définitions de livrables généralement produits avec l'application d'une méthode agile :

- le **carnet de produit** (*Backlog, Product Backlog*) : liste de toutes les spécifications du produit à réaliser. Les spécifications (éléments du carnet ou *product backlog items [PBI]*) sont généralement rédigées sous l'une des deux formes suivantes :
  - les scénarios utilisateurs (*Users Stories*) : spécifications rédigées de manière à décrire un besoin utilisateur à combler[1] ;
  - les cas d'utilisation (*Use Cases*) : spécifications rédigées de manière à décrire une fonctionnalité du système à construire[2].
- la **définition de terminé** (*Definition of Done*) : définition de la qualité de livraison. Cette même définition est utilisée en complément aux conditions de succès des éléments du carnet de produit pour qualifier chaque incrément de logiciel, à chaque itération. Elle assure une qualité uniforme de la part de tous les membres de l'équipe de projet, car ils sont engagés à respecter cette définition.

2.3 L'ADOPTION PARTIELLE DES PRATIQUES

La précédente section a donné une liste générale des pratiques agiles. Certaines organisations, intéressées seulement par quelques bénéfices de l'agilité, décident

1. Pour plus de détails, nous vous invitons à consulter l'ouvrage *User Stories Applied: For Agile Software Development* par Mike Cohn, Addison-Wesley, 2004.
2. Pour plus de détails, nous vous invitons à consulter l'ouvrage, *Rédiger des cas d'utilisation efficaces* par Alistair Cockburn, Eyrolles, 2001.

d'adopter l'agilité une pratique à la fois, plutôt que l'ensemble des pratiques d'une méthode. Ces organisations adoptent donc partiellement les pratiques agiles.

Chaque méthode agile est un ensemble cohérent de pratiques. Une organisation ayant choisi une adoption partielle a le défi de sélectionner l'agencement des pratiques permettant d'atteindre les objectifs d'amélioration visés. En effet, certaines pratiques sont dépendantes ou préalables à d'autres pratiques et il est important de bien en connaître l'ensemble pour augmenter l'efficacité d'une adoption partielle.

L'adoption partielle des pratiques est traitée par l'ouvrage *Agile Adoption Patterns : A Road to Organizational Success* d'Amr Elssamadisy. Cet auteur reconnaît qu'il n'est pas nécessaire d'adopter l'ensemble des pratiques agiles d'un seul trait et qu'il est possible de le faire de manière progressive. Selon les objectifs choisis, il propose des lots de pratiques et un ordre d'adoption. Par exemple, la Figure 2.1, illustre les pratiques que l'auteur propose d'adopter lorsque l'objectif est l'amélioration du temps de mise en marché. Le sens des flèches indique l'ordre d'adoption des pratiques.

Une organisation désirant adopter la pratique du *Développement de tests automatisés* devrait commencer par appliquer la pratique de la *Livraison fréquente* ainsi que celle de l'*Intégration continue*. Négliger l'implantation des pratiques préalables pour des questions de coûts ou d'économie de temps diminue grandement l'efficacité des tests automatisés.

Pour atteindre pleinement son objectif, l'organisation doit adopter l'ensemble des pratiques proposées à la Figure 2.1. Bien que cela représente plusieurs pratiques, le modèle exclut tout de même les pratiques n'étant pas directement liées à l'atteinte de l'objectif. C'est le cas notamment de la pratique du *Binômage* ou de la *Mêlée quotidienne*, quoi qu'utiles, elles ne sont pas liées à l'objectif de réduction du *Temps de mise en marché*.

2.4 LA COMBINAISON DES MÉTHODES

À l'opposé de l'adoption partielle, il est possible d'appliquer plus d'une méthode à la fois. La combinaison de méthodes est un concept largement répandu dans les organisations agiles. Les méthodes, même si elles ont des objectifs différents, sont souvent compatibles l'une avec l'autre. Certaines de leurs pratiques, telle la *Mêlée quotidienne*[1], se recoupent. Il est donc possible de les conjuguer pour obtenir une méthode spécifique à une organisation. Ainsi, il est fréquent d'unir les méthodes Scrum, XP et Lean : Scrum pour la gestion de projet incrémentale, XP pour se doter de bonnes pratiques d'ingénierie logicielle, Lean pour augmenter l'efficacité des processus de développement de l'organisation.

La combinaison peut mener à choisir que les pratiques les plus attrayantes de chacune des méthodes et ainsi créer une méthode agile particulière à une organisation.

1. La mêlée quotidienne est nommée *Daily Scrum* dans la méthode *Scrum* et *Stand-up meeting* dans la méthode XP. Mais c'est essentiellement la même pratique.

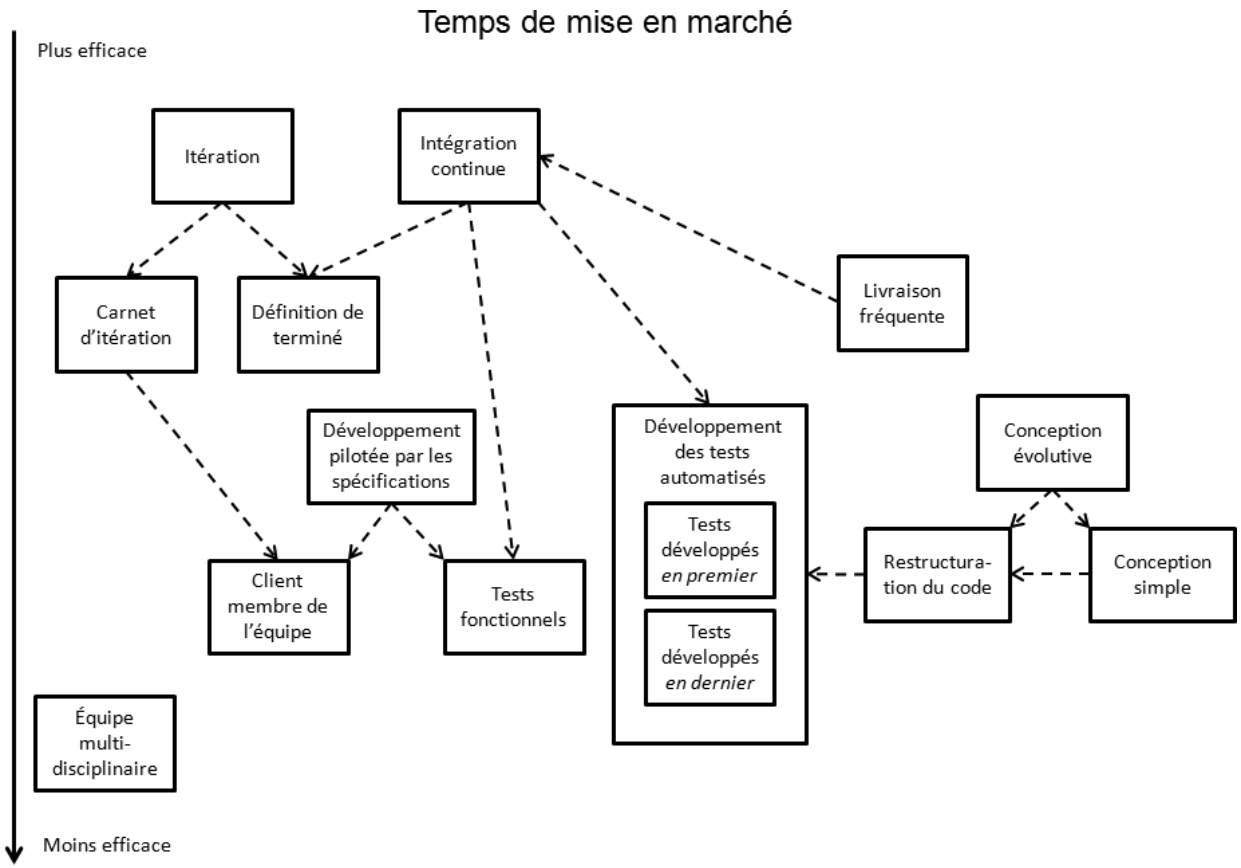


Figure 2.1 — Ensemble proposé de pratiques agiles à adopter pour améliorer le temps de mise en marché[a].

a. Tiré de l'ouvrage *Agile Adoption Patterns: a Road to Organizational Success* par Amr Elssamadisy, Addison-Wesley, 2008.

Nous recommandons d'être prudent au moment de modifier, adapter, ou combiner les pratiques des méthodes agiles. En particulier lorsque l'abandon de pratiques est envisagé. Pour les organisations débutant avec l'agilité et ne maîtrisant pas les valeurs et les principes du manifeste, l'abandon d'une pratique est souvent un prétexte pour cacher un symptôme rendu visible.

Retour d'expérience

Il arrive parfois que les organisations que nous accompagnons nous questionnent sur l'intérêt de tenir une rétrospective à toutes les itérations, prétextant que c'est une rencontre coûteuse parce qu'elle monopolise périodiquement l'ensemble de l'équipe de développement. Pourtant, l'un des principes agiles est l'amélioration continue, soutenue par la tenue régulière de rétrospectives.

Lorsqu'une organisation est prête à abandonner cette pratique, je l'invite à reconsidérer sa décision ou alors de proposer une alternative aussi efficace pour obtenir des résultats d'introspection. Habituellement, elle choisit de conserver la pratique, n'ayant pas trouvé de meilleure alternative.

Mathieu

Nous conseillons souvent aux équipes débutantes de résister à la tentation d'adapter les méthodes disponibles dans l'industrie et de plutôt se questionner sur les raisons les poussant à adopter une méthode agile. Si toutefois une organisation est convaincue de son choix, nous lui conseillons alors de consulter la section de l'adoption partielle et le manifeste agile pour vérifier que cet abandon ne se fait pas au détriment d'une ou plusieurs autres pratiques.

2.5 LES ENTENTES DE TRAVAIL

L'adoption d'une méthode agile peut-être fragile. Appliquer l'agilité suppose beaucoup de rigueur et de discipline. Cela n'est pas toujours facile lorsque les embûches surviennent. Lorsqu'elles subissent de la pression, les équipes ont le réflexe de retourner à leurs anciennes façons de faire qu'elles maîtrisent parce qu'elles les ont appliquées pendant plusieurs années. Pour demeurer agiles, les équipes ont besoin d'avoir confiance en elles et confiance aux résultats des pratiques agiles qu'elles adoptent.

Une façon d'augmenter la confiance et la rigueur des équipes est de définir les principes du manifeste comme des ententes de travail introduites par l'adoption de l'agilité. Nous définissons *entente de travail par* une proposition, un énoncé qui n'est pas encore été démontré mais qui est jugé vrai jusqu'à preuve du contraire. Les ententes de travail aident à mobiliser les individus pas encore convaincus par les bienfaits et les avantages de l'agilité.

Sans ententes de travail, les membres de l'organisation peuvent se diviser autour de l'agilité : les personnes en faveur du *Manifeste agile* risquent d'être perçues comme des puristes tandis que les autres seront perçues comme des réfractaires.

Nous proposons ici de constater comment les ententes de travail peuvent prévenir et démentir les arguments des sceptiques de l'agilité.

Argument n° 1 : Il n'est pas nécessaire de livrer le système à toutes les itérations, car nous n'aurons encore rien d'intéressant à présenter à nos utilisateurs.

Il est intéressant de livrer des incréments de logiciel à la fin de chaque itération lorsqu'il est entendu que chaque incrément n'est vraiment valable que s'il est démontré dans un environnement final. La livraison incrémentale permet de s'exercer régulièrement à faire toutes les activités nécessaires pour livrer en production. Cela suppose que chaque incrément de logiciel disponible est comparable à celui de la production et que la vélocité pour produire de tels incréments est la véritable capacité de l'équipe de projet.

Une seconde entente est que la vraie vélocité d'une équipe comprend toutes les disciplines et les activités nécessaires pour livrer le système en production. Ne pas livrer à toutes les itérations, c'est d'admettre que certaines activités et risques ne seront rendus visibles qu'à la toute fin du projet. Ces découvertes viendront immanquablement allonger la période de stabilisation en fin de projet.

Argument n° 2 : Il n'est pas possible de démarrer le projet, l'analyse n'est pas assez avancée.

Pour démarrer un projet sans une analyse complète, il doit être entendu qu'il n'est pas possible de tout prévoir correctement et qu'il est nécessaire de faire des preuves de concept ou des incréments de logiciel pour ajuster la conception et avoir un logiciel le plus adapté possible aux besoins.

Autrement dit, commencer un projet avec une analyse détaillée, c'est prendre le risque de devoir reprendre des travaux d'analyse ou de conception lorsque d'autres disciplines comme la programmation ou les tests fonctionnels en exposeront les manques ou les erreurs.[1]

Argument n° 3 : L'équipe de développement a besoin d'un chef de projet. Les développeurs n'ont pas l'intérêt ou les compétences pour mener un projet à bien.

Les équipes auto-organisées sont pertinentes lorsqu'il est entendu que les personnes les plus proches de la réalisation sont les mieux placées pour prendre des décisions.

Une seconde entente associée est qu'il préférable que les experts soient des conseillers au service des équipes plutôt que des validateurs. Il est difficile pour les

1. Pour plus de détails sur cet aspect, se référer au chapitre *4 Architecture incrémentale*.

experts de valider le travail des équipes sans nuire à leur vélocité. L'auto-organisation prévient le fait qu'une équipe de projet devienne dépendante de plusieurs experts.

Conseil : Il se peut que l'équipe ne soit pas prête à s'auto-organiser et que les gestionnaires soient inquiets de la laisser à elle-même. Dans ce cas, l'accompagnement par un coach peut leur offrir une période de préparation et de transition.

Argument n° 4 : Il n'y a pas d'intérêt à réunir des développeurs .Net et COBOL ensemble. La nature de leur travail est trop différente.

La constitution d'équipes pluridisciplinaires est justifiée lorsqu'il est entendu que la contribution de tous les spécialistes est nécessaire pour produire un incrément de logiciel complet.

Ne pas réunir toutes les spécialités d'une même équipe, c'est accepter que le produit doive passer entre les mains de plusieurs équipes pour pouvoir être complété. Cela peut provoquer des contraintes d'intégration et des ajustements entre les équipes qui auront des dépendances et des attentes les unes envers les autres.

Argument n° 5 : Il n'est pas nécessaire que le client soit un membre à temps plein de l'équipe de projet.

L'affectation d'un responsable d'affaires au sein des équipes de projet est justifiée lorsqu'il est entendu qu'il est la meilleure personne pour recueillir les besoins des parties prenantes et pour vérifier que le produit réponde bien aux attentes des utilisateurs finaux.

Autrement, c'est accepter que quelqu'un du département des TI doive simuler ce rôle, et cela implique qu'il devra passer du temps avec les responsables d'affaires pour éviter de développer une mauvaise solution logicielle. Il faudra aussi avoir conscience que ce représentant a plus de chances de faire une mauvaise interprétation des besoins et des priorités qu'un expert du domaine d'affaires.

2.6 PRÉPARER L'ADOPTION DE L'AGILITÉ

Une fois qu'une organisation a défini les raisons la poussant à adopter l'agilité, elle pourrait décider de limiter l'adoption à un premier projet[1]. Au-delà de la définition des ententes de travail, d'autres aspects sont à considérer pour maximiser les chances de réussite de l'introduction l'agilité.

1. Nous nommons cette stratégie de transition *progressiste*. L'ensemble des stratégies de transitions organisationnelles sont couvertes par le *chapitre 11 Gouvernance*.

2.6.1 Les critères pour choisir un projet candidat

Certains projets sont de meilleurs baromètres pour tester une transition agile. L'idée est d'accumuler une expérience suffisamment pertinente pour adapter le processus organisationnel à l'intention des autres projets. Dans son ouvrage *Succeeding with Agile : Software Development using Scrum*, Mike Cohn identifie quatre aspects à considérer pour choisir un projet servant d'étalon à une transition, soit : la durée du projet, l'appui du management, la criticité du projet et la taille de l'équipe mandatée.

La durée du projet

Choisir un projet trop court n'est pas idéal pour vérifier la bonne application de l'agilité. Il faut plusieurs itérations, voire quelques mois, pour entrevoir les bonnes et les mauvaises tendances qu'un projet pourrait prendre.

Par exemple, si l'équipe ne développe pas avec le souci d'écrire un code clair et souple, elle pourrait ne pas connaître de problème de vélocité dû à la difficulté naissante d'introduire de nouvelles fonctionnalités dans un code mal conçu. Pour observer ce dysfonctionnement, il faut quelques mois.

Les projets trop longs auront un autre effet. Par exemple, un projet de dix-huit mois risque de ne pas rencontrer de problème de stabilisation et de livraison avant longtemps. Cela n'aidera pas à vérifier que la définition de terminé est complète et couvre tous les aspects de développement et de livraison de votre organisation. Avec un projet long et pour limiter cet effet, il peut être envisageable de le découper en plusieurs livraisons.

Un projet de six à neuf mois est probablement un bon candidat pour permettre d'expérimenter un cycle complet de développement dans un délai raisonnable.

L'appui du management

L'adoption de la philosophie de travail agile se voit facilitée lorsque les différentes directions du projet comprennent les enjeux et la portée de l'agilité, et acceptent d'adapter leur méthode de gestion, dans les bons moments comme dans les moins bons.

Il est à noter que l'appui de la direction ne doit pas se limiter aux TI. Les approches agiles préconisent une collaboration étroite entre les responsables d'affaires et ceux de TI. Cela est vrai des projets de développement jusqu'à la sphère du management[1].

La criticité du projet

L'idéal est de choisir un projet représentatif, suffisamment grand pour être crédible et relativement critique pour donner une chance à la transition de s'opérer.

Il est tentant de prendre le projet le plus simple possible pour éviter que l'adoption d'une approche agile ait trop d'impact sur le fonctionnement de l'organisation. Cela n'est malheureusement pas représentatif des conséquences réelles que l'approche aura

1. Les impacts sur le rôle des gestionnaires est couvert au chapitre *11 Gouvernance*.

sur le processus organisationnel. Dans une mesure raisonnable, il est préférable de choisir un projet rencontrant les problèmes faisant l'objet des objectifs d'amélioration visé par l'adoption de l'agilité.

À l'opposé, il faut éviter de choisir les projets sur lesquels l'organisation fonde le plus d'espoir. Sinon, les attentes placées envers le projet risquent de miner l'initiative de la transition. Cela n'est cependant pas toujours possible parce que justement, l'agilité est peut-être le moyen choisi pour redresser l'organisation. Lorsqu'un projet extrêmement critique ne peut être évité, il faut s'assurer que tous les intervenants comprennent que l'agilité n'est pas seulement une tentative, mais qu'elle représente un objectif stratégique pour l'organisation.

La taille de l'équipe mandatée

Une équipe agile ne devrait pas être trop grande pour permettre l'auto-organisation et la bonne communication. Les praticiens de la méthode Scrum proposent généralement que la taille des équipes soit comprise entre cinq et dix membres. Au-delà de ce nombre, il faut penser à diviser l'équipe et introduire des mécanismes de synchronisation et d'intégration.

Autant que possible, il est préférable d'éviter ce genre de complexité dans les premières équipes agiles. L'adoption de la philosophie de travail agile peut-être difficile en soit et, d'expérience, il est préférable de comprendre cette philosophie avant de se lancer dans une mécanique multi-équipe.

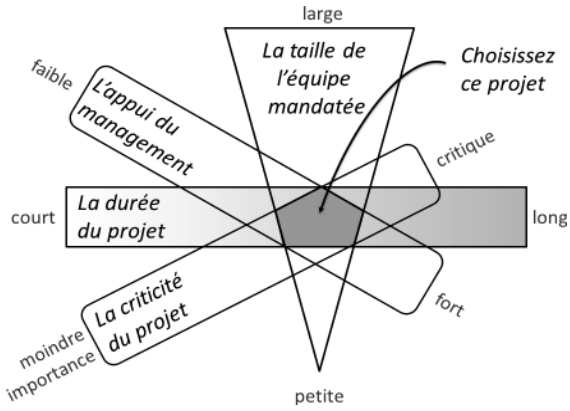


Figure 2.2 — Les critères pour choisir un projet candidat au démarrage de l'adoption de l'agilité (extrait et traduit du livre *Succeeding with Agile: Software Development using Scrum*, de Mike Cohn, Addison-Wesley Professional, 2009).

2.6.2 Projet en cours ou nouveau projet ?

Les projets en cours comme les nouveaux projets peuvent être candidats à l'introduction de l'agilité. Il peut sembler préférable de sélectionner un nouveau projet pour introduire une transition agile. C'est effectivement un bon candidat, parce qu'il

est plus facile de commencer avec des nouvelles façons de faire que de modifier des habitudes de travail.

Cependant, c'est probablement l'un de vos projets en cours ou passé qui est à l'origine de l'intérêt de l'agilité dans votre organisation. De ce point de vue, les projets en cours sont aussi d'excellents baromètres et il y aurait beaucoup de valeur à les choisir pour introduire votre transition.

Dans ce cas, prenez le temps de vous arrêter, de vérifier le travail restant à faire, de construire un carnet de produit, de découper le reste du projet en itérations. Bref, poursuivez le projet, comme s'il était nouveau, en appliquant d'abord les différents ateliers de démarrage[1].

Pour terminer, nous vous invitons à prendre connaissances des attentes suscitées par les projets candidats, ainsi que des impacts sur les projets adjacents, indépendamment de la façon dont vous les avez sélectionnés. Par exemple, connaissez-vous les attentes que peuvent avoir les parties prenantes envers le projet ? Ont-ils la même compréhension que vous des objectifs d'amélioration, du nouveau processus de développement, des conditions gagnantes à réunir, des facteurs de risque et des nouvelles hypothèses de travail ? L'écoute, expliquer pourquoi et la communication intensive sont, parmi d'autres, des techniques typiques de la gestion du changement aidant à réduire le nombre de personnes réticentes à la mise en place de l'agilité.

En résumé

Il est souhaitable d'identifier les raisons qui poussent une organisation à adopter une approche agile pour l'aider à atteindre des objectifs d'amélioration. À partir de ces objectifs, le choix des pratiques et des méthodes sera mieux ciblé et elles seront alignées avec les objectifs d'amélioration.

Toutes les méthodes agiles répondent au même manifeste, mais chacune se concentre sur un aspect différent du développement logiciel. Par exemple, Scrum est une méthode axée sur la gestion de projet tandis que XP est une méthode axée sur les pratiques d'ingénierie logicielle.

Dans l'ensemble, les pratiques agiles visent à réduire les difficultés du développement logiciel. Cependant, certains sous-ensembles sont plus efficaces que d'autres pour traiter des enjeux spécifiques.

Les pratiques des méthodes agiles ont toutes leur utilité et il faut s'interroger quand une organisation souhaite en supprimer certaines. C'est peut-être le signe qu'elle essaie de cacher un problème rendu visible par les nouvelles pratiques.

Une fois les objectifs d'amélioration connus, les méthodes et les pratiques sélectionnées, il faut choisir le projet avec le meilleur potentiel de réussite. Pour faire le bon choix, il est préférable de réunir les conditions de succès autour du projet et de s'entendre sur les hypothèses de travail. Cela évite les remises en question stériles et une division de l'organisation entre les puristes agiles et les réfractaires.

1. Se référer au chapitre *3 Démarrage de projet agile* pour plus d'informations sur ces ateliers.

3

Démarrage d'un projet agile

Objectif

Dans ce chapitre, vous découvrirez pourquoi la phase de démarrage est essentielle au succès d'un projet en mode agile, comment se découpe cette phase en différents ateliers, quels sont les livrables attendus, qui devrait y prendre part et quelles sont les conditions d'arrêt de façon à pouvoir commencer la première itération d'un projet.

Mise en situation : François, Vincent et le démarrage d'un projet

Vincent est le développeur senior de son équipe de développement. Avec François, un chargé de projet, il participe à une formation d'introduction au rôle de Scrum Master. Tous les deux sont employés dans une compagnie d'assurance et ils sont affectés à un prochain projet pilote ayant pour objectif de vérifier les bienfaits des méthodes agiles dans l'organisation et de vérifier si ces méthodes sont compatibles avec les processus de développement de la compagnie.

Le projet consiste à développer un outil pour permettre aux clients corporatifs de faire eux-mêmes les demandes des réclamations et d'inscrire leurs employés à l'assurance collective. La compagnie a terminé son plan de mise en marché et elle est prête à démarrer dans les deux prochaines semaines. La compagnie a investi dans une équipe de projet composée d'un responsable de produit, un Scrum Master, un analyste d'affaires et quatre développeurs qui travailleront au développement du système pendant neuf mois. Un mois supplémentaire a été prévu pour la période de stabilisation et la livraison du système.

En tant que chargé de projet, François, avec l'aide de Vincent, doit préparer le démarrage du projet : réunir l'équipe pour les ateliers de démarrage, prendre connaissance de l'énoncé de projet et planifier la réalisation du cahier des charges.

Tous deux ont compris que le carnet de produit est un aspect important de la planification dans la méthode Scrum. Mais comment faire pour transformer le cahier

des charges en carnet de produit ? Comment construire un plan de livraison à partir d'un carnet de produit ? Et à quel moment les spécialistes devront-ils être invités ?

François : On devrait peut-être se construire des ateliers à partir des préalables de la méthode Scrum : un carnet de produit priorisé, un responsable de produit, une équipe de projet, un Scrum Master, puis s'entendre sur une définition de terminé et la durée des itérations.

Vincent : Je suis d'accord. Scrum est une méthode incrémentale et il faut éviter de faire l'analyse détaillée complète du cahier des charges pour commencer. Je propose qu'on limite nos ateliers de démarrage à l'objectif « d'être prêt à planifier notre première itération ».

François : Limitons nos efforts et démarrons une première itération Mais mon expérience de chargé de projet me recommande d'en faire un peu plus. Je pense que l'organisation serait plus confortable si notre projet diffusait un plan de livraison avant le démarrage.

Vincent : Ça me va, mais comment faire un plan de livraison sans faire une analyse détaillée ?

3.1 LA PORTÉE DES ACTIVITÉS DE DÉMARRAGE

Les méthodes agiles préconisent une approche incrémentale, soit le développement de logiciel par incréments. Pour être en mesure de livrer une solution logicielle de manière incrémentale, il faut que les disciplines du développement logiciel soient exécutées en parallèle plutôt qu'en séquence. Chaque itération comporte donc un certain lot d'activités de planification, d'analyse, de conception, d'écriture de code, de tests et de livraison.

Si avec une approche agile les activités sont distribuées à l'intérieur du projet, il peut être déduit que les activités de démarrage requerront moins d'efforts. Nous ne disons pas que la période de démarrage est inutile sous une approche agile. Nous affirmons cependant qu'il est possible de revoir les livrables préalables et limiter les efforts requis dans le but de commencer le développement le plus tôt possible.

3.1.1 Cycle de projet agile

La période de démarrage est la première étape d'un projet de développement. Selon la méthode *Agile Unified Process*[1] (Agile UP), le cycle de développement d'un projet se divise en quatre phases :

- l'évaluation[2];
- l'élaboration ;
- la construction ;
- la transition.

1. http://www.ambysoft.com/unifiedprocess/agileUP.html consulté en mars 2011.
2. Traduction libre du terme anglais « *inception* »

La phase d'**évaluation** a pour objectif de produire les livrables relatifs à la portée du projet, comme la charte de projet, l'architecture de haut niveau, un cahier de spécifications et un plan de livraison initial. Au terme de cette phase, les parties prenantes devraient être en mesure d'évaluer si un projet doit se poursuivre ou s'arrêter. Lorsqu'un projet franchit l'acceptation de la phase d'évaluation, l'objectif de la phase de l'**élaboration** est de gérer les éléments de risque du projet : le chemin critique fonctionnel, l'architecture, l'environnement de développement. Avec des risques mitigés, la phase de **construction**, se consacrera à la réalisation productive du système. Lorsque le projet est prêt pour une livraison finale, le projet sera alors dans sa phase de **transition,** qui comprend la stabilisation et le déploiement.

La Figure 3.1 illustre comment, avec une approche agile, la proportion des efforts requis pour chacune des disciplines du développement logiciel n'est pas la même à chacune des phases du cycle du projet. Par exemple, les efforts de modélisation sont concentrés dans les phases d'évaluation et d'élaboration alors que la programmation est concentrée à la phase de construction. Cependant, aucune des disciplines n'est dédiée qu'à une seule phase et les efforts leur étant consacrés varient selon le déroulement du projet.

La représentation graphique de la distribution des efforts par discipline démontre que l'objectif de la phase de l'évaluation ne conclut pas les activités d'analyses. Il faut donc accepter, qu'au terme de la phase de l'évaluation, il restera plusieurs incertitudes liées au projet comme :

- Comment utiliser les technologies mal maîtrisées, parce qu'elles sont nouvelles ou désuètes ?
- Comment utiliser les nouveaux outils de développement ?
- Quels sont les inconnus et les points de litige du domaine d'affaires ?
- Comment collaborer avec les équipes externes et intégrer leurs travaux au reste du projet ?

Ces incertitudes nécessiteront de prévoir du temps à l'intérieur des différentes phases pour développer des preuves de concept qui viendront valider les hypothèses brutes de l'analyse, de la conception et de la programmation. Dans le cas où ces hypothèses s'avèrent invalides, un projet itératif permet d'introduire des alternatives sans nuire à la planification. Le principe est de repousser la plupart des décisions au « dernier moment responsable », évitant ainsi d'accumuler un inventaire d'hypothèses dont l'exactitude n'est pas prouvée au départ d'un projet.

L'accumulation d'un inventaire de travaux qui ne sont pas validés est la première cause de l'effet tunnel. Celui-ci survient lorsque la visibilité sur le déroulement du projet se perd pendant plusieurs semaines, voire des mois. Le résultat de l'effet tunnel est souvent le développement d'un produit inutile, mal ciblé, ne répondant pas aux besoins du client et demandant un échéancier et un budget plus grands qu'initialement prévu. Pour éviter ce risque, une synchronisation est effectuée au terme de chacune des itérations, permettant de démontrer l'avancement du produit et d'ajuster le plan, même pendant les phases d'évaluation et d'élaboration.

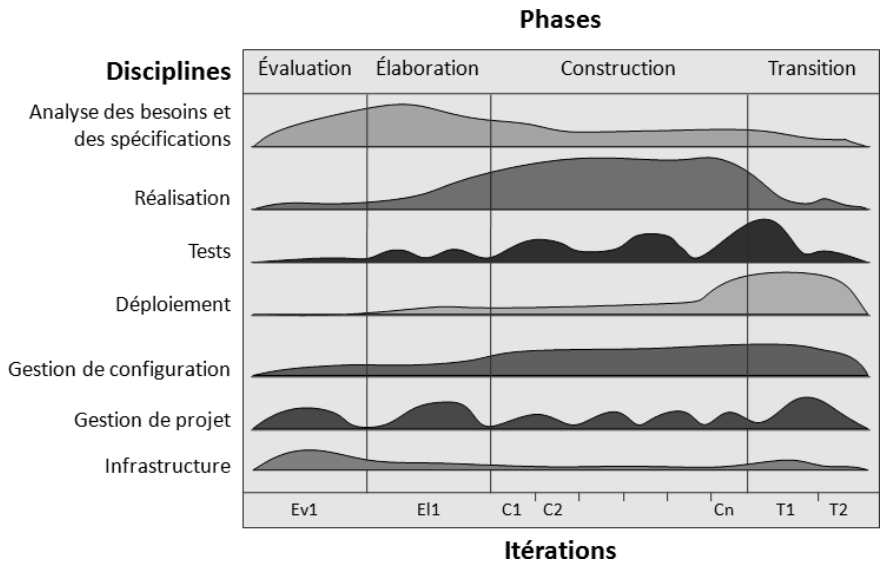


Figure 3.1 — Disciplines à travers les phases de l'*Agile Unified Process*.

Certains affirment qu'il est difficile de distribuer l'analyse et la conception à l'intérieur des itérations, parce qu'elles sont trop courtes pour démontrer des résultats. À notre avis, les équipes de projets ont souvent cette perception lorsqu'ils traitent des problèmes très complexes. Ils ont alors le réflexe d'étirer les ateliers de démarrage pour répondre à cette complexité. Plus souvent qu'autrement, la complexité n'est pas la racine du problème. Ce serait plutôt le manque d'expérience à aborder les projets par petits morceaux et par ordre de priorité.

3.1.2 Les objectifs des activités de démarrage

Les objectifs des ateliers de démarrage sont donc les mêmes que ceux de la phase de l'évaluation, soit :

- fournir suffisamment d'information aux parties prenantes responsables d'autoriser la poursuite du projet ;
- produire les livrables requis pour pouvoir démarrer la phase de l'élaboration.
- C'est également une période pour :
- **communiquer la vision**, les objectifs, les enjeux, les contraintes, les risques et les conditions de succès du projet ;
- **dispenser de la formation** aux équipes sur les processus et les méthodes utilisés dans le projet, même à ceux connaissant déjà le sujet. L'objectif est que tous les contributeurs partagent la même connaissance ;
- **poursuivre l'analyse et la conception** : prévoir des moments individuels pour que les intervenants recueillent des informations manquantes et fassent quelques

preuves de concepts pour réduire les incertitudes inhérentes aux premières itérations ;

- **construire une équipe de projet** : les méthodes agiles prônent la collaboration et l'engagement. Un bon objectif serait que tous les contributeurs connaissent leur rôle et leurs responsabilités au commencement du projet.

Selon notre expérience, ces objectifs de démarrage sont atteints à travers quatre grandes activités, soit :

- la formation d'une équipe de démarrage ;
- la rédaction d'une charte de projet ;
- l'élaboration d'un carnet de produit ;
- la préparation d'un plan de livraison.

3.2 FORMER UNE ÉQUIPE DE DÉMARRAGE

Qui devrait participer aux activités de démarrage ? Le plus de gens possible ! Les architectes, les analystes, le marketing, les experts d'affaires, l'ergonome, les développeurs, le vice-président, etc. Un projet de développement est un investissement et il est important que tous les intervenants sachent de quoi il en retourne. Pour augmenter l'engagement de tous les collaborateurs autour du succès du projet, il est important qu'ils interagissent tôt ensemble et qu'ils s'informent, avec le moins d'intermédiaires possibles, des enjeux et des objectifs du projet. Une grande participation augmente le sentiment d'appartenance général envers le projet et encourage les initiatives au quotidien qui ne dépendront plus d'un seul responsable.

Chaque intervenant ayant un rôle différent, certains participants pourront quitter avant la fin des ateliers. Seule l'équipe de réalisation, le coach d'équipe et le client[1] sont tenus de rester jusqu'à la toute fin de la période de démarrage. L'idée n'est pas d'inviter n'importe quel curieux ou de faire perdre du temps à l'organisation, mais plutôt de réunir les intérêts et les préoccupations de toutes les personnes concernées par le projet.

Si vous êtes préoccupés que le nombre d'individus réduise l'efficacité des rencontres, nous vous invitons à sélectionner les meilleurs intervenants pour chaque activité et à diviser les participants en sous-groupes selon le sujet des ateliers.

1. Pour un souci de simplification, nous utiliserons le terme « *client* » dans ce chapitre lorsque nous voulons décrire la personne experte du domaine d'affaires responsable de la spécification et de la planification du projet. Nous utiliserons le terme « *coach* » lorsque nous voulons décrire la personne qui accompagne et aide l'équipe à devenir responsable de la gestion de projet de manière efficace et collaborative.

Retour d'expérience : Identifier la contribution des participants

En tant que coach agile, j'ai régulièrement l'occasion d'animer les ateliers de démarrage des équipes qui débutent avec les approches agiles. Quand je me retrouve devant dix personnes et plus, j'aime bien commencer avec l'activité du « diagramme des rôles ».

Je demande à tous les participants d'inscrire leur nom sur un Post-it™ et de le placer sur un diagramme des rôles comme celui de la Figure 3.2.

Les participants choisissent de placer leur nom du côté affaires ou du côté informatique, selon leur expertise et la nature de leur rôle. Ceux qui s'engageront d'itération en itération placent leur nom au centre, car ils sont membres de l'équipe de projet.

Ce simple exercice me permet de discuter avec les intervenants sur leur contribution envers le projet. Souvent, l'exercice révèle des zones floues :

– certains ne savent pas où placer leur nom, parce qu'ils ne connaissent toujours pas leur rôle,

– certains placent leur nom sur une frontière, parce qu'ils voudraient s'engager, mais savent bien qu'ils en seront incapables avec toutes leurs affectations,

– parfois, l'équipe informatique est complètement absente, ou encore le client n'est pas présent.

Lorsque les rôles ne sont pas clairs, j'ajoute aux objectifs des ateliers de démarrage des discussions pour clarifier les rôles. S'il manque des membres de l'équipe de projet, j'ajourne les ateliers le temps de remédier à la situation.

Bien entendu, on peut utiliser n'importe quel autre type d'atelier pour clarifier la position de chaque rôle, mais je ne veux pas laisser planer le doute sur le rôle et les responsabilités des participants pendant les ateliers de démarrage.

Mathieu

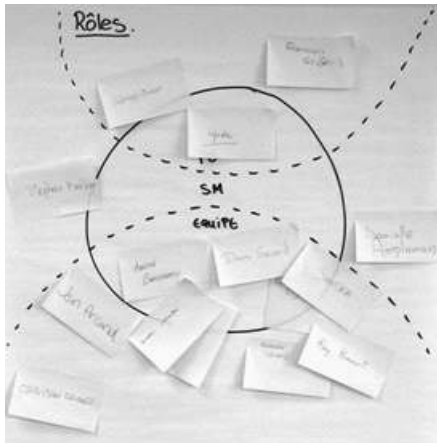


Figure 3.2 — Exemple de diagramme des rôles.

3.2.1 Les dirigeants

Les dirigeants sont un bon exemple d'intervenants ne participant pas toujours à la totalité des ateliers. Ils sont invités pour communiquer la vision et les enjeux au reste de l'équipe de projet. Il ne faut pas sous-estimer le message fort que la présence d'un dirigeant apporte aux ateliers de démarrage. Nous en avons rencontré plusieurs réticents à participer aux activités de démarrage, jugeant que leur présence serait interprétée comme de l'ingérence. Nous sommes d'avis que si un projet est stratégique, au point d'investir des sommes considérables dans son développement, les dirigeants se doivent de partager les attentes et l'intérêt suscités par le projet. Cette présence sera perçue comme du respect envers l'équipe et un signal fort de l'importance du projet.

Nous aimons commencer les ateliers en leur donnant la parole : leur discours n'est jamais bien long et il résume la raison d'être du projet. Parce que leur discours est souvent concis et ciblé, il n'est pas rare qu'on y entende LA phrase qui résume le projet dans la charte de projet. Si aucun dirigeant n'est disponible, le client se chargera d'introduire le projet au début des ateliers.

3.2.2 Le client, le responsable de produit, les experts d'affaires

Valeur agile : Collaboration avec le client plutôt que négociation de contrat.

Le rôle du client dépasse celui de référent expert du domaine d'affaires. Son rôle est critique au succès du projet et ses responsabilités s'approchent de celles du gestionnaire de produit traditionnel. C'est lui qui doit planifier l'exécution du projet, collaborer étroitement et fréquemment avec l'équipe de projet et diffuser l'état d'avancement du produit au reste de l'organisation. Les ateliers de démarrage sont un moment propice pour valider que le client assigné est le bon candidat. Quelques symptômes de dysfonctionnement ne mentent pas :

- le client n'est pas disponible pour se présenter aux ateliers de démarrage ;
- il n'est pas en mesure d'expliquer la portée du projet ;
- il ne peut prendre de décision sans systématiquement aller vérifier avec ses pairs ;
- sa vision se limite à son expérience personnelle et ne tient pas compte des intérêts des autres parties prenantes du produit.

Un bon candidat est en mesure de s'affirmer lorsqu'il est interrogé au sujet des enjeux d'affaires du projet.

Même si le client se doit d'être un expert d'affaires le plus polyvalent et autonome possible, il n'est pas nécessaire qu'il soit un expert de tous les domaines d'affaires touchés par le projet. D'autres experts peuvent s'adjoindre à lui pour compléter ses compétences et ainsi faciliter la fluidité des ateliers de démarrage. Parmi les experts d'affaires, nous pensons notamment aux utilisateurs experts, aux analystes d'affaires ou métier, aux pilotes de systèmes et aux membres du service à la clientèle.

3.2.3 Les architectes, les analystes, les ergonomes...

L'analyse et la conception occupent une grande place des activités de démarrage et c'est pourquoi nous invitons les spécialistes à participer et à communiquer au reste de l'équipe l'architecture et la modélisation envisagées pour le projet. Cependant, les spécialistes doivent être vigilants à ne pas faire une analyse préliminaire détaillée du système à ce stade-ci. Au terme des ateliers de démarrage, les dossiers de conception et d'architecture doivent être juste suffisants[1] pour être en mesure d'estimer les exigences brutes du produit : un diagramme de contexte, les processus d'affaires documentés à haut niveau et le schéma des données d'affaires peuvent suffire. Si les besoins d'analyse sont plus grands, rappelons qu'il est envisagé d'étaler les travaux au-delà de la phase d'élaboration du projet.

> **Principe agile** : Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto-organisent.

Les approches agiles préconisent que la conception et l'analyse d'un système relèvent de la responsabilité de l'équipe de projet. Même si l'objectif est de former des équipes multidisciplinaires, cela ne veut pas dire que les spécialistes sont inutiles et exclus. Les ateliers de démarrage vont permettre de clarifier le rôle des spécialistes au sein de l'équipe de projet et mesurer l'impact de l'adoption de l'agilité sur leur méthode de travail[2].

3.2.4 L'équipe de projet

Pendant la période du démarrage, l'équipe de projet est amenée à comprendre le projet, découvrir le matériel des spécialistes, estimer les exigences du produit et participer à la rédaction de la définition de terminé.

Le démarrage d'un projet coïncide souvent avec la constitution d'une nouvelle équipe de projet. La participation active des membres de l'équipe aux ateliers catalyse la construction d'une équipe performante. Les ateliers sont également une occasion pour vérifier que tous les membres de l'équipe ont une motivation à collaborer. Si un développeur pense que sa contribution n'est pas utile, il faut lui faire comprendre qu'au terme des ateliers, lui et ses coéquipiers doivent se sentir suffisamment en confiance pour s'engager auprès du client et réaliser une première itération.

3.2.5 Le coach, chargé de projet ou Scrum Master

L'agilité fait la promotion des équipes autonomes et des individus engagés. Une équipe de projet travaillant étroitement avec son client est un groupe autonome et pluridisciplinaire pouvant mener son projet au succès.

---

1. En référence au terme « *just enough* » de la littérature anglaise sur le Lean.
2. Le chapitre *4 Architecture incrémentale* décrit plus précisément l'impact de l'agilité sur le rôle de l'architecte.

Le coach d'équipe a les mêmes préoccupations que les chargés de projet classiques. Mais plutôt que de contrôler les activités, comme le ferait un chargé de projet, il fait réfléchir son équipe pour qu'elle ne perde pas de vue les valeurs de l'agilité et les objectifs du projet. Au commencement du projet, il anime son équipe et s'assure qu'elle ne s'éparpille pas dans le temps et qu'elle se concentre sur les objectifs des activités de démarrage.

3.2.6 Les autres intervenants

L'impact qu'aura un projet sur le reste de l'organisation est souvent négligé. Plus l'organisation est grande, plus l'impact d'un projet sur les individus est grand. Les personnes impactées auxquelles nous pensons comprennent notamment les utilisateurs finaux, les pilotes de systèmes, les formateurs, le service à la clientèle, l'équipe marketing, les opérateurs ou encore les administrateurs des applications.

Une bonne façon de réduire la résistance au changement causé par le développement et la livraison d'un nouveau produit est de les impliquer tôt dans le processus et de discuter avec eux du moment le plus propice pour qu'ils démarrent leurs interventions avec l'équipe de projet. S'il est contre-productif d'inviter ces personnes pendant les ateliers, il serait au moins bon de les identifier et de prendre la peine d'aller les rencontrer pour avoir cette discussion le plus tôt possible.

3.3 LIMITER LES EFFORTS REQUIS

> Réaction d'un détracteur :
> « *Notre projet est évalué à 2 000 jours-personnes, avec une équipe de 15 développeurs. Allons-nous vraiment faire participer toute l'équipe aux ateliers de démarrage* ? »

Nous reconnaissons qu'une quinzaine de personnes rassemblées pour concevoir une application pendant cinq jours représente une somme considérable en salaires. C'est pour cela qu'il est important que les activités de démarrage se fassent en évitant le gaspillage en temps et en efforts. Voici quelques exemples des types de gaspillage pouvant se produire pendant la période de démarrage :

- une architecture détaillée, qui n'est pas validée par un incrément de logiciel fonctionnel, source de l'effet tunnel ;
- un plan de livraison précis basé sur une capacité d'équipe approximative ;
- la rédaction des spécifications pour des fonctionnalités qui, au final, ne seront jamais produites.

Dans la plupart des organisations où nous intervenons, nous avons l'habitude de tenir les ateliers de démarrage en 5 à 10 jours ouvrables. Quelquefois moins, lorsque tous les participants sont bien préparés ou que le projet n'est pas trop complexe. Mais il ne faut pas prendre cette durée comme étant absolue et limiter les activités à l'extrême en risquant de démarrer le projet dans le chaos. Tous les projets n'ont pas la même

complexité. Il est difficile d'imaginer qu'un projet répondant à un sujet aussi sensible et complexe que l'atterrissage en toute sécurité d'un avion puisse démarrer sans une période d'analyse et de préparation plus longue.

Un indicateur pour s'assurer que l'effort consacré aux ateliers de démarrage est suffisant, est de vérifier qu'au terme de la période, l'équipe de projet est en mesure de s'engager dans une première itération. Une période de démarrage dont la durée est plus grande que celle d'une itération normale est un indice que trop d'efforts y sont consacrés. Selon nous, la production de livrables essentiels est un autre indicateur que les activités de démarrage sont efficaces et concluantes, parce que ce sont des livrables permettant de démarrer un projet avec un haut niveau de confiance :

- une charte de projet ;
- un carnet de produit ordonné et priorisé ;
- un document d'architecture de haut niveau ;
- une définition de terminé ;
- un plan de livraison.

Itération 0 : Plusieurs praticiens de l'agilité nomment la courte période de démarrage l'*Itération 0* (ou *Sprint 0*). Cette itération se distingue des autres parce qu'elle implique souvent plus d'intervenants que la seule équipe de projet et qu'elle est plus courte qu'une itération normale.
Cette appellation est cependant controversée, parce qu'elle n'est pas documentée dans la littérature de Scrum, la méthode agile la plus populaire à ce jour. Pourquoi faire une itération spéciale avant les autres itérations ? Pourquoi ne pas commencer immédiatement les vraies itérations ? Effectivement, il n'y a pas vraiment de logique à nommer cette période « itération », encore moins de lui attribuer un « 0 »[1]. Ne soyons pas puristes, retenons simplement que l'itération 0 est reconnue dans la communauté agile comme une période de préparation au démarrage des projets.

3.4 RÉDIGER UNE CHARTE DE PROJET

La charte de projet est un document décrivant de manière concise la vision, les objectifs, les leviers et les enjeux du projet.

3.4.1 Pourquoi ?

Le document est un guide qui aide l'équipe de projet à prendre les bonnes décisions en cours de projet, en ciblant les objectifs et les conditions de succès.

1. Anecdote : cela dit, nous trouvons encore plus étrange l'appellation « Itération -1 », décrite dans l'un des billets de Scott Ambler (http://www.drdobbs.com/architecture-and-design/209902719, consulté en mars 2011).

3.4.2 Combien ?

La charte de projet doit être concise pour être facile à consulter, voire facile à afficher. Elle peut facilement tenir sur une ou deux pages (voir).

3.4.3 Comment ?

Voici un modèle de structure de charte de projet :

- Une section pour la vision : une phrase résumant la raison d'être du projet. Une bonne formule pour décrire une vision est celle du test de l'ascenseur (voir l'encadré ci-après).
- Une section pour les objectifs : une liste des objectifs à atteindre avec la réalisation du projet.
- Une section pour les mesures de succès : une liste des critères et des indicateurs permettant de vérifier que les objectifs ont été atteints.
- Une section pour de la matrice des compromis : un indicateur de la marge de manœuvre pour chacun des leviers permettant de contrôler la tenue du projet. Certains projets ont une date de livraison inflexible, d'autre ne peuvent pas faire de compromis sur la portée. Un projet où aucun compromis n'est possible sur tous les trois leviers d'ajustement (portée, investissement, date de livraison) est un projet sans marge de manœuvre pour gérer les risques.
- Une section pour les facteurs de risque : les obstacles potentiels ou avérés du projet. L'identification des obstacles permet de les traiter ou de faire une demande d'aide à l'organisation pour ceux dont l'équipe de projet n'a aucun contrôle.

Test de l'ascenseur

Imaginons que vous montiez dans un ascenseur en même temps que le président d'une très grande entreprise. Le président reconnaît votre visage mais ne vous connaît pas. Curieux, il vous demande sur quoi vous travaillez. Vous avez maintenant dix étages pour vous présenter !

Votre présentation pourrait avoir la forme suivante : Suite à l'opportunité <Y>, le projet <X> s'adresse à <public> pour leur offrir un <bénéfice>. Contrairement à <concurrent>, notre projet se distingue par <particularités>.

En pratique, cela pourrait donner ceci : Le projet « Télé sur le web » s'adresse à nos abonnés et vise à réduire nos coûts d'exploitation en nous rendant éligibles à la réduction de taxes offertes aux diffuseurs de chaînes de télévision publique. De plus, cette fonctionnalité nous permettra de vendre un accès aux chaînes privées dont nous sommes les propriétaires exclusifs sur toutes les plates-formes de nos abonnés : téléphone, ordinateur, décodeur numérique.

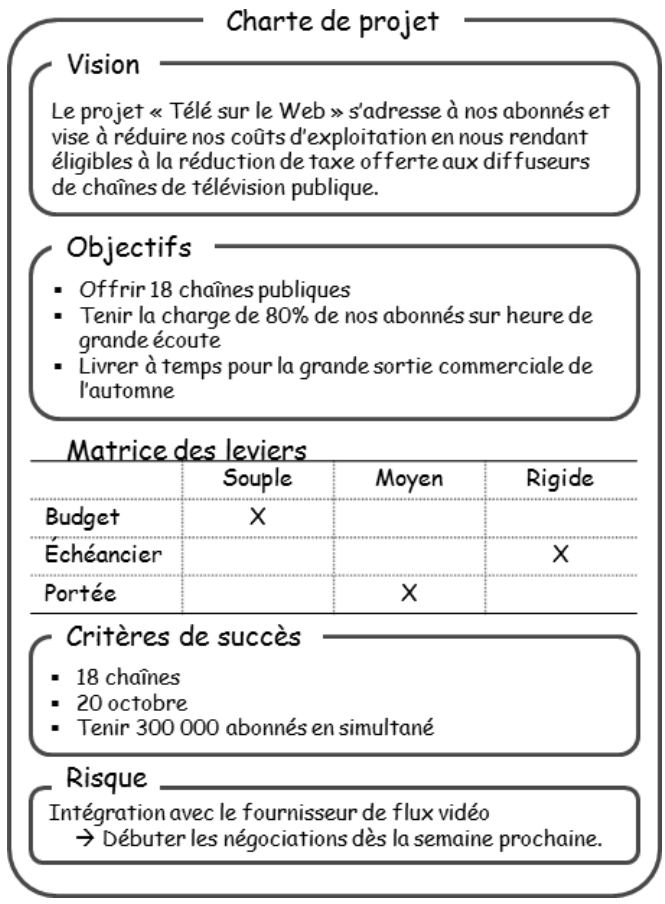


Figure 3.3 — Exemple d'une charte de projet.

Retour d'expérience

Sur un de nos projets, notre levier le plus flexible était le budget, car le client avait un budget quasi illimité. Aux trois quarts du projet, nous nous sommes aperçus que le projet risquait de ne pas atteindre la portée désirée à la date prévue. Cependant, nous ne savions pas comment utiliser l'argent pour remédier à la situation. Embaucher de nouvelles personnes dans le projet si près de la date finale semblait être contre-productif.

Nous avions appris une leçon. Il ne suffit pas d'identifier les leviers disponibles : certains leviers ont besoin d'une stratégie pour être efficace, et nous aurions dû en prendre conscience pendant nos ateliers de démarrage.

Mathieu

3.5 CONSTRUIRE UN CARNET DE PRODUIT

Un carnet de produit est une liste de besoins d'affaires ordonnés selon leur valeur d'affaires, dont chaque item est décrit sous la forme d'une exigence ou d'une spécification.

- **Pourquoi ?**
  Pendant les ateliers de démarrage, le carnet de produit sert à recueillir toutes les exigences qu'il est prévu de développer pendant le projet. Pendant le développement du projet, le carnet de produit sert d'intrant aux séances de planification. À la fin de chacune des itérations, l'état d'avancement du projet est mesuré en comparant l'état du produit et le contenu restant du carnet de produit.
- **Combien ?**
  Idéalement, il faut suffisamment d'items détaillés pour que le client puisse planifier ses deux premières itérations. L'équivalent de deux itérations permet au client de ne pas être au dépourvu si un item particulier ne peut pas être planifié parce qu'il doit être raffiné ou qu'un préalable manquant empêche son développement.
  Au-delà de deux itérations, les items risquent d'être détaillés prématurément et de devoir être modifiés pour répondre aux besoins changeants du projet. Pire, certains items peuvent être détaillés d'avance et au final ne jamais être développés.
- **Comment ?**
  Le client est responsable de l'entretien du projet. Il utilise le carnet de produit pour gérer, rédiger, prioriser les exigences du produit. Il peut ajouter ou retirer autant d'éléments que nécessaire. Cependant, la stratégie est de concentrer la conception et le raffinement des exigences par ordre de priorité : une stratégie limitant les efforts inutiles.
  Il faut cependant apporter une certaine nuance à cette stratégie, puisque la valeur ajoutée n'est pas le seul élément à surveiller lors de la planification d'un projet, il faut également penser à l'utilisabilité. Prenons l'exemple d'un système de commerce électronique. Même si les méthodes de paiement sont des fonctionnalités à forte valeur ajoutée, tant que le système ne permet pas aux clients de faire des achats, la fonctionnalité de paiement en ligne demeure inutile. Autre exemple, une application de recherche d'emploi qui ne permet pas à un employeur d'afficher des offres d'emploi et de rencontrer des candidats aura une valeur très faible. Une simple annonce dans un journal imprimé aurait déjà plus de valeur pour un employeur. Développer une solution fonctionnelle devrait être le premier objectif de la planification et la priorisation par valeur ajoutée devrait en découler.

Nous vous proposons dans cette section quelques pratiques et conseils pour vous aider à la construction du carnet de produit :

- le *story mapping* pour construire un carnet de produit selon des couloirs fonctionnels ;

- des **pratiques d'ordonnancement** adaptées aux stratégies de planification incrémentale et la gestion du risque ;
- des **pratiques pour calculer la valeur d'affaires** des items du carnet de produit, adaptées aux stratégies de planification de fonctionnalités à grande valeur ajoutée ;
- la rédaction des exigences, adaptée au développement itératif et incrémental d'une solution logicielle.

3.5.1 Story mapping

Le *story mapping*[1] est un exercice proposé par Jeff Patton permettant de construire un carnet de produit en identifiant les couloirs fonctionnels de la solution. Les couloirs fonctionnels sont des processus d'affaires complets, comme la « recherche de candidats pour une offre d'emploi » dans l'exemple du site de recherche d'emploi.

La première étape de l'atelier est de prioriser tous les éléments du carnet de produit selon l'ordre chronologique logique de l'utilisation. Par exemple, dans un système de recherche d'emploi, la publication d'offres d'emploi se produit habituellement avant que les chercheurs d'emploi puissent soumettre leur candidature.

La seconde étape est de classer les items selon leur indice de criticité. La criticité se définit comme étant une combinaison de la fréquence d'utilisation de la fonctionnalité et de son importance. Toujours avec l'exemple d'un site de recherche d'emploi, joindre un *curriculum vitae* (CV) au moment de la candidature est probablement plus critique que d'être capable d'éditer son CV en ligne.

La cartographie représentant la combinaison de la priorité et de la criticité permet d'identifier les couloirs fonctionnels du carnet de produit. La première tranche de la cartographie représente probablement l'ensemble minimal de fonctionnalités permettant à tous les types d'utilisateurs du système d'utiliser le produit dans sa forme la plus simple. La construction du carnet de produit autour des couloirs fonctionnels facilite la construction d'une solution utile (voir la Figure 3.4).

3.5.2 Ordonner le carnet de produit

Dans son livre *Agile Estimating and Planning*[2], Mike Cohn, propose un modèle de priorisation prenant en compte la valeur d'affaires et la gestion du risque. Dans la Figure 3.5, ce modèle propose de commencer par les éléments les plus risqués ayant également la plus forte valeur ajoutée (1er quadrant, en haut à droite). C'est la stratégie qui est adoptée lorsque le premier objectif est la réalisation d'une solution fonctionnelle. Le développement d'un premier couloir fonctionnel, comme proposé dans l'atelier *Story Mapping* est un bon moyen d'atteindre cet objectif.

1. http://www.agileproductdesign.com/blog/the_new_backlog.html, consulté en mars 2011.
2. Cohn, Mike, *Scrum : Agile estimating and planning*, Prentice Hall, 2005.

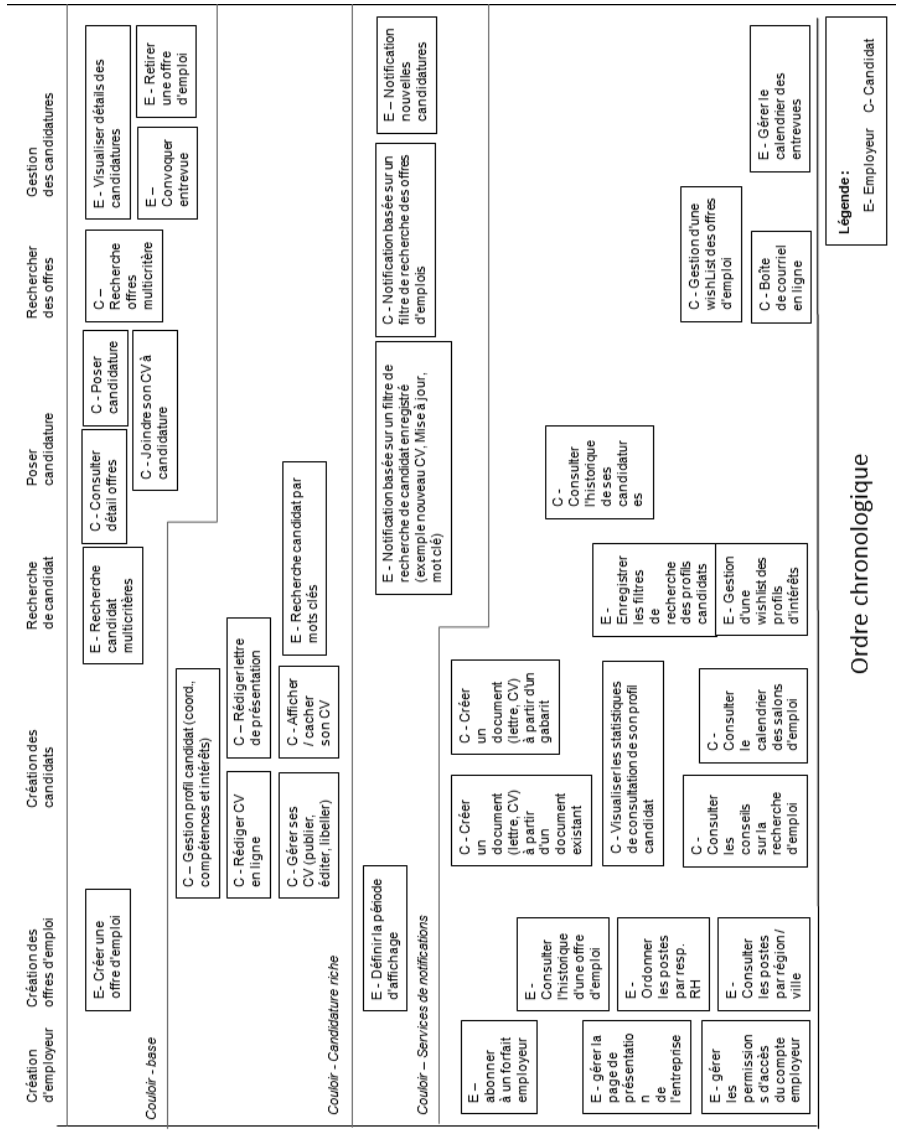


Figure 3.4 — Exemple du résultat final de l'exercice du *Story Mapping*.

Le 2e quadrant (coin inférieur droit) est celui des éléments à forte valeur ajoutée mais dont les facteurs de risque sont faibles. Ces éléments sont intéressants puisqu'ils ont un fort retour sur investissement : beaucoup de valeur pour peu de risque.

Le 3e quadrant (coin inférieur gauche) représente les éléments ayant moins de valeur d'affaire et étant moins risqués. Ces éléments seront probablement les derniers à être développés.

Ce modèle ne propose pas de planifier les éléments risqués ayant peu de valeur d'affaires. En effet, si on veut éviter de développer des fonctionnalités qui seront peu ou pas utilisées, il faudrait éviter de planifier ces éléments même s'ils faisaient partie de la portée initiale du projet[1].

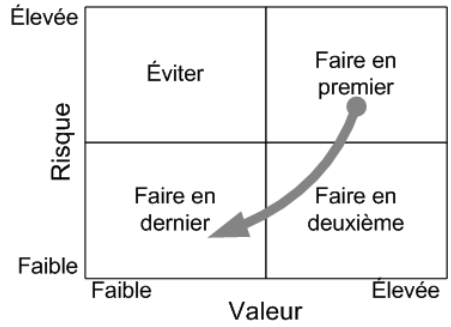


Figure 3.5 — Stratégie de priorisation des exigences.

3.5.3 Calculer la valeur d'affaires des items du carnet

Toutes les organisations TI visent à livrer de la valeur pour leurs clients, sans qu'elle soit systématiquement quantifiée ou rendue visible à tous. Les approches agiles incitent explicitement les équipes à planifier le développement logiciel par

ordre d'importance et insistent pour que chaque livraison ajoute une valeur d'affaires au produit développé de manière incrémentale. Toutefois, les approches agiles ne préconisent pas de technique particulière pour mesurer la valeur d'affaires des fonctionnalités d'un projet.

Il est possible, au début d'une transition agile, que les clients ne quantifient pas la valeur d'affaires des fonctionnalités qu'ils priorisent. Typiquement, ils ne l'ont jamais fait, soit parce qu'ils ne savent pas comment faire, soit que ce n'est tout simplement pas dans leurs habitudes de travail. Les méthodes agiles prônant la transparence, il est normal que cette transparence s'applique aussi à la valeur d'affaires attribuée à chacun des items d'un carnet de produit. C'est pourquoi il est fortement encouragé

1. Rappelons la troublante statistique du rapport du Standish Group qui mentionne que 45 % des fonctionnalités développées ne sont jamais utilisées. La planification par priorité permet de diminuer ce pourcentage, en éliminant les éléments à faible valeur ajoutée.

d'accompagner les responsables de produit à quantifier cette valeur et de l'utiliser dans les activités de gouvernance. Ainsi, les gestionnaires, dirigeants et responsables de produits seront motivés à fournir et maintenir cette valeur.

Dans certaines grandes entreprises où la valeur d'affaires de chaque fonctionnalité et caractéristique est quantifiée depuis de nombreuses années, on va jusqu'à en quantifier le rendement. Cette analyse permet de vérifier l'exactitude de la quantification de la valeur, servant ainsi à alimenter et améliorer le processus de quantification. Nous ne disons pas qu'il faille se rendre là dès le début d'une transition agile, mais seulement qu'avec le temps, les organisations désirant augmenter la maturité à leur processus en ce sens vont jusqu'à mesurer le rendement réel.

Même convaincue de la pertinence de cette mesure, une organisation n'ayant jamais mesuré la valeur d'affaires des fonctionnalités d'un projet peut se sentir démunie face à cette pratique de gestion. Bien qu'il soit possible de déterminer la valeur d'affaires de manière intuitive, nous présentons ici un résumé de quelques pratiques permettant de mesurer la valeur d'affaires des fonctionnalités et caractéristiques.

La valeur financière

Nous avons travaillé avec plusieurs compagnies d'assurance où des actuaires étaient en mesure de calculer financièrement le rendement de chacune des fonctionnalités prévues d'un système. Sachant qu'il existe peu de domaines d'affaires bénéficiant des services d'un actuaire, nous vous présentons un résumé des étapes à suivre pour calculer cette valeur :

1. **Identifier les revenus potentiels** : il existe plusieurs types de catégorisation possibles pour mesurer les sources de revenus que peut engendrer le développement d'une fonctionnalité. Elle peut être une source de nouveau revenu en attirant de nouveaux clients. Elle peut être une source de revenu supplémentaire si elle permet d'augmenter le prix qu'un client existant est prêt à investir dans les services. Elle peut être un manque à gagner si l'absence de la fonctionnalité ne retient pas les clients existants. Et finalement elle peut être une source d'économie si elle permet d'améliorer l'efficacité des opérations.
2. **Comparer les revenus aux coûts** : une fois estimés, les revenus peuvent être comparés avec les coûts requis à la livraison de la fonctionnalité. Le ratio entre les revenus et les coûts est déjà un indicateur permettant de prioriser les fonctionnalités à développer.
3. **La valeur nette actualisée (VAN)**[1] : les revenus estimés pour une fonctionnalité ne sont pas les mêmes lorsqu'un projet s'étend sur une longue période. En effet, les revenus qu'il n'est pas possible de générer aujourd'hui représentent un manque à gagner sur des investissements. Le calcul de la VAN permet de tenir compte de l'amortissement des revenus à travers le temps.

1. VAN, traduction de l'anglais *Net Present Value, NPV.*

4. **Le retour sur investissement (RSI)**[1] : Tandis que la VAN permet de connaître le retour net d'un investissement dans le futur, le RSI permet de savoir le taux de retour sur investissement à travers le temps. Autrement dit, plus le RSI est grand, plus vite le capital sera remboursé et plus rapidement l'investissement rapportera des revenus.

L'évaluation du potentiel monétaire est un élément de planification très efficace et sans ambiguïté. Cependant, il n'est pas simple car ses calculs demandent des compétences en mathématiques financières.

La valeur de la satisfaction

Le modèle de Kano[2] permet de mesurer la valeur d'affaires des fonctionnalités selon la satisfaction qu'elles apportent aux utilisateurs. Voici un résumé des étapes à suivre pour déterminer la satisfaction potentielle des fonctionnalités :

1. À partir d'un questionnaire à l'intention des utilisateurs, évaluer le degré de satisfaction que l'utilisateur retirerait si la fonctionnalité était disponible et le degré d'insatisfaction si la fonctionnalité était absente.
2. Le résultat des deux questions pour chaque fonctionnalité permet de catégoriser le degré de satisfaction selon trois groupes :

   a) les **fonctionnalités essentielles** sont considérées comme implicites par les utilisateurs, qui retirent peu de satisfaction lorsqu'elles sont présentes et beaucoup de frustration lorsqu'elles sont absentes ;

   b) les **fonctionnalités exprimées** sont proportionnelles à leur présence. Plus un produit contient de fonctionnalités exprimées, plus le client est prêt à payer cher pour l'utiliser ;

   c) les **fonctionnalités attrayantes** procurent beaucoup de satisfaction aux utilisateurs qui n'avaient aucune attente particulière à leur sujet.

L'évaluation de la satisfaction est à la portée de tous et permet de faire une évaluation fiable de la valeur d'affaires des fonctionnalités d'un carnet de produit. Elle permet en outre de connaître le minimum de fonctionnalités attendues par les clients, de savoir pour quelles fonctionnalités les utilisateurs sont prêts à payer plus et comment attirer des clients enthousiastes à l'aide de fonctions attrayantes.

Pour plus de détails et de référence sur le calcul de la valeur financière et du degré de satisfaction, nous vous invitons à consulter l'ouvrage *Agile Estimating and planing*[3]. Évidemment ce ne sont pas les seuls moyens d'évaluer la valeur d'affaires des fonctionnalités d'un produit, mais cet ouvrage de Mike Cohn a popularisé ces techniques auprès d'une portion significative de praticiens agiles.

1. RSI, traduction de l'anglais *Return on Investment, ROI*.
2. Source : http://fr.wikipedia.org/wiki/Mod%C3%A8le_de_Kano, consulté en mars 2011.
3. *Scrum : Agile estimating and planning* de Mike Cohn, Prentice Hall, 2005.

3.5.4 Détailler les exigences

Une fois que le carnet de produit décrit grossièrement l'ensemble de toutes les exigences du projet, les items sont détaillés dans l'ordre des priorités. La description des items du carnet de produit peut prendre plusieurs formes : scénario utilisateur) (*user story*), cas d'utilisation (*use case*) ou autres. Il faut s'assurer que la forme choisie se prête bien à la planification incrémentale. L'acronyme INVEST (Indépendant, Négociable, Valeur d'affaires, Estimable, *Small* [petit], Test fonctionnel), que Mike Cohn propose dans son ouvrage *User Stories Applied*[1] rappelle les qualités des items se prêtant bien au développement agile. Pour plus de détails sur la rédaction du carnet de produit en mode agile, vous pouvez consulter les livres de Mike Cohn et Alistair Cockburn, *User Stories Applied for Agile Software Development*[2] et *Rédiger des cas d'utilisation efficaces* [3].

3.6 FIXER LA DÉFINITION DE TERMINÉ

La définition de « terminé » est une description sans équivoque décrivant tout ce qui doit être fait pour considérer un élément du carnet de produit comme terminé. Elle sert de référence pour les membres de l'équipe de projet pour assurer que tout ce qui est terminé répond aux mêmes critères de qualité, peu importe qui a contribué à sa réalisation. La définition de terminé représente le contrat de qualité entre le client et l'équipe de réalisation.

3.6.1 Pourquoi ?

> **Principe agile** : Un logiciel fonctionnel est la meilleure unité de mesure de la progression du projet.

Les approches agiles mesurent l'avancement des projets par la proportion de fonctionnalités d'un produit opérationnelles à un moment donné. Selon ce principe, une fonctionnalité qui serait seulement documentée dans un dossier d'analyse ne pourrait être compilée dans l'état d'avancement d'un projet, puisqu'elle n'est toujours pas opérationnelle au sein du produit. Pour devenir fonctionnelle, les autres disciplines de développement comme le code, l'intégration et les tests doivent avoir été complétées.

Avec une définition de terminé, les membres de l'équipe s'entendent sur toutes les activités requises pour compléter les items du carnet de produit. Lorsqu'un membre annonce qu'un item est terminé, tous ont la confiance qu'il ne reste pas de travail à accomplir pour être en mesure de la livrer.

1. *User Stories Applied : For Agile Software Development*, de Mike Cohn, Addison-Wesley Professional, 2004.
2. *Scrum: Agile estimating and planning* de Mike Cohn, Prentice Hall, 2005.
3. *Rédiger des cas d'utilisation efficaces* de Alistair Cockburn, Eyrolles, 2001.

Durant les ateliers de démarrage, l'équipe devra estimer les éléments du carnet de produit pour être en mesure de rédiger un plan de livraison. Au moment d'estimer, la définition de terminé ne devrait pas être négociable. Alors que la portée d'une exigence peut varier, les activités de développement devraient être fixes. Cette définition de la qualité peut faire varier la capacité d'une équipe à livrer des incréments de logiciel fonctionnel, mais elle n'affecte pas la valeur des estimations.

3.6.2 Combien ?

Plus la définition de terminé couvre tout le travail à faire pour livrer un item du carnet de produit, plus les chances de découvrir de mauvaises surprises seront réduites lorsque le produit sera prêt à être déployé en production. Il faut donc répertorier toutes les activités de développement de l'organisation et les inscrire à la définition de terminé du projet. Cela couvre les activités de rédaction jusqu'à celles de stabilisation. D'ailleurs, une définition complète et respectée réduira le temps nécessaire pour stabiliser les produits avant leur mise en production.

3.6.3 Comment ?

Les éléments de la définition ne devraient pas laisser place à interprétation. Par exemple, un élément qui s'intitulerait « Les tests unitaires sont écrits », ne permet pas de vérifier si le niveau de qualité défini a été rencontré ou pas. Chacun des membres peut avoir une conception différente de ce que tester le code représente.

Pour prévenir cette mauvaise interprétation, Amr Elssamadisy, dans son ouvrage *Agile Adoption Patterns*[1], propose d'adopter des objectifs dont les éléments qui sont « SMART », c'est-à-dire **S**pécifique, **M**esurable, **A**tteignable, pe**R**tinent (***r**elevant*) et limité dans le **T**emps. En ajoutant un seuil de couverture de tests par module de l'application, l'élément « Les tests unitaires sont écrits » pourrait devenir « Les tests unitaires couvrent à 100 % le code du module de service et le code des modules de la couche du domaine ». Ce genre d'élément ne laisse plus place à interprétation, et la vérification d'une fonctionnalité se basera sur des critères mesurables.

Pour poursuivre votre réflexion à propos de la définition de terminé, vous pouvez vous référer aux ouvrages *Scrum, le guide pratique de la méthode agile la plus populaire*[2] de Claude Aubry et *Agile Estimating and Planning*[3] de Mike Cohn.

3.7 BÂTIR UN PLAN DE LIVRAISON

Rédiger un plan de livraison suite à l'adoption d'un processus de planification itératif est un véritable défi. Comment estimer la taille d'un projet alors que les items du

1. *Agile Adoption Patterns* de Amr Elssamadisy, Addison-Wesley Professional, 2008.
2. *Scrum : Le guide pratique de la méthode agile la plus populaire* de Claude Aubry, Dunod, 2010.
3. *Scrum : Agile estimating and planning* de Mike Cohn, Prentice Hall, 2005.

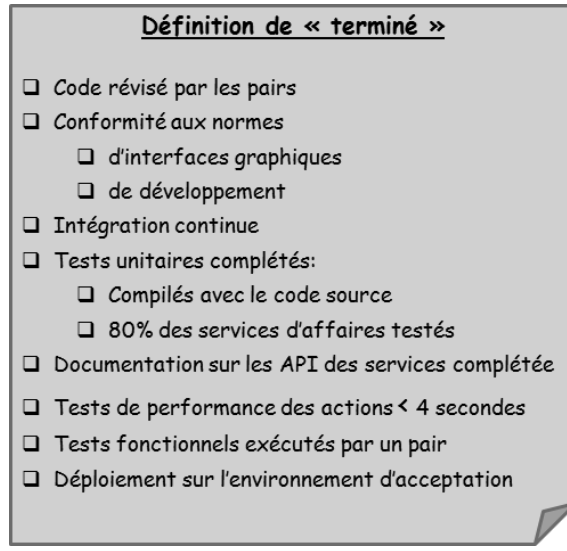


Figure 3.6 — Exemple d'une définition de terminé.

carnet de produit ne sont pas tous détaillés ? En effet, à la section du « Carnet de produit », nous avons mentionné que les exigences doivent rester brutes et qu'elles seront raffinées au fur et à mesure de la progression du projet. Par contre, sans détail, il est difficile de les estimer de façon précise.

Pour résoudre cette ambiguïté, les estimations faites au démarrage d'un projet sont basées sur le meilleur des connaissances et de l'expérience de l'équipe de projet. D'ailleurs une estimation est, par définition, fausse. C'est une approximation faite à partir d'une une hypothèse posée. Et comme avec toutes les hypothèses, elle doit être vérifiée.

L'estimation au démarrage sert à mesurer l'ampleur du projet et pour tenter de vérifier si le projet est réaliste, malgré que cette projection comporte une grande incertitude.

3.7.1 Pourquoi ?

Parce qu'il est requis de vérifier si le projet est réaliste avec les ressources disponibles. C'est un constat qui doit être fait à la fin de chaque itération et il est attendu que le plan de livraison soit mis à jour à la même fréquence[1].

1. Dans le chapitre 6 *Gestion de projet*, nous discutons de la manière de réviser le plan de livraison pendant le déroulement du projet.

3.7.2 Combien ?

Le plan de livraison devrait tenir lisiblement sur une seule page.

3.7.3 Comment ?

Voici un exemple d'atelier pour produire un plan de livraison. Imaginons que nous ayons un carnet de produit. À ce stade, nous devrions avoir un carnet de produit ordonné par importance et dont les items les plus prioritaires sont détaillés. Ces éléments détaillés doivent pouvoir s'estimer sans trop de difficulté, surtout s'ils respectent les critères INVEST.

Pour estimer le reste du carnet de produit, il n'est pas nécessaire d'en faire le détail complet. Les éléments détaillés et estimés peuvent servir à évaluer rapidement le reste du carnet de produit, par analogie. En comparant, chaque item du carnet de produit avec celles estimées, il est possible de faire une estimation relative. Cette estimation pourra être révisée lorsque nous serons prêts à raffiner le reste du carnet. Mais pour l'instant, notre objectif est d'évaluer l'ampleur de notre projet. Lorsque toutes les spécifications auront été estimées par comparaison, nous aurons un carnet de produit ordonné et estimé.

En estimant la capacité de l'équipe pendant une itération, le nombre d'itérations nécessaires pour compléter le carnet de produit en entier pourra être estimé. Ou encore, si la date de fin du projet est fixée, nous pourrons déduire combien de fonctionnalités pourront être livrées pour cette date.

En résumé

Les activités au démarrage d'un projet sont sensiblement les mêmes :
– la rédaction d'une charte de projet ;
– la constitution d'une équipe de projet ;
– la rédaction d'un carnet de produit ;
– la préparation d'un plan de livraison.

Il faut savoir répondre à ces objectifs dans une courte période si l'on veut éviter de produire une analyse détaillée, qui risque d'accumuler du gaspillage, cause principale de l'effet tunnel.

La condition d'arrêt des ateliers devrait « être prête à itérer ». Pour être prêt à itérer, il faut avoir, au minimum, un carnet de produit priorisé et estimé, ainsi qu'une équipe dédiée à la réalisation.

Certains livrables comme la charte de projet, le plan de livraison et la définition de terminé sont très utiles pour pouvoir faire le suivi et la gestion de projet.

Les ateliers de démarrage ont avantage à réunir tous les intervenants du projet pour encourager la collaboration et s'entendre sur le mode d'intervention de tous les acteurs du projet. Ceci est autant vrai pour l'équipe de projet dédiée que pour les spécialistes qui gravitent autour du projet.

4

Architecture incrémentale

Objectif

Dans ce chapitre, vous distinguerez l'analyse préliminaire détaillée de la conception incrémentale préconisée par les approches agiles. Vous découvrirez quelles sont les qualités recherchées d'un bon architecte logiciel, pas seulement son savoir-faire, mais aussi ses compétences de savoir-être. Puis, vous explorerez en détail comment, avec une approche incrémentale, conjuguer l'analyse et la planification des enjeux techniques avec celles des besoins d'affaires.

Mise en situation : Josiane et l'architecture traditionnelle

Josiane exerce dans le domaine du développement logiciel depuis près de 30 ans. Durant les quinze dernières années, elle élabore des architectures logicielles de plus en plus riches dans des contextes de plus en plus complexes. S'appuyant sur des processus traditionnels, elle a toujours complété ses documents d'architecture détaillée avant de les présenter à l'équipe de développement, avec le but de prévenir la révision des documents. Ces documents d'architecture de haut niveau sont détaillés et révisés par un comité d'architecture où chacun des membres se concentre sur des aspects spécifiques : définition des fonctionnalités et attribution à un système spécifique, choix technologiques (plateformes, langages et outils), et structure des programmes. L'élaboration de l'architecture détaillée demande de 6 à 18 mois, retardant d'autant le début de la réalisation.

Josiane commence alors une nouvelle affectation dans une société d'état qui entreprend une vaste refonte de l'ensemble de ses applications développées il y a plus de vingt ans, dans une technologie n'étant plus supportée aujourd'hui et pour laquelle il devient difficile de trouver une main-d'œuvre qualifiée. Le directeur des TI nouvellement arrivé

lui explique que sa société a pris la décision d'adopter les façons de faire agiles et qu'elle devra s'adapter à ces nouvelles méthodes. C'est-à-dire qu'elle devra maintenant :
– Définir une architecture sommaire tout juste suffisante pour permettre aux équipes de démarrer rapidement ;
– Détailler le reste de l'architecture au fur et à mesure du déroulement du projet ;
– Collaborer étroitement avec les équipes de développement, équipes qui feront émerger certaines parties de l'architecture ;
– Repenser la structure et le cycle de révision des documents d'architecture.
Bien sûr, un coach agile s'adjoindra à Josiane pour l'aider dans sa nouvelle mission.
« *Mais ce n'est pas possible !* » se dit-elle. « *Ça ne va jamais marcher leur truc ! Les développeurs n'ont jamais suffisamment la vue d'ensemble pour penser correctement aux détails. Cela va créer une quantité monstre de travail à reprendre à chacune des nombreuses impasses que le projet rencontrera !* ».
C'est dans cette attitude fermée au changement que Josiane commença son affectation.

4.1 L'ARCHITECTE LOGICIEL

Il est reconnu au titre d'architecte logiciel une expertise dans des domaines précis de la conception et de la réalisation des systèmes. L'architecte traite des préoccupations complexes du développement logiciel comme la sécurité, la robustesse, la performance, l'intégrité, l'intégration, la conformité, etc. Les organisations confèrent parfois à ce titre une certaine notoriété et autorité s'expliquant par la reconnaissance de cette expertise.

L'objectif de l'architecte est de trouver des solutions qui répondent le mieux aux besoins du client et qui conviendront au contexte informatique de son organisation. C'est pourquoi il doit s'investir à connaître les produits, les langages, les *patterns*, les technologies du marché et celles de l'organisation pour pouvoir les utiliser au bon moment et à bon escient. C'est à partir de cette expertise qu'il devient possible de concevoir correctement des solutions matérielles et logicielles. C'est un rôle qui peut faire la différence dans le développement d'une solution de qualité.

Bien que leur expertise soit un facteur de succès des projets de développement, certains architectes sont de grands réfractaires à l'adoption des approches agiles. Mike Cohn, dans son ouvrage *Succeeding with Agile : Software Development Using Scrum* identifie deux raisons majeures à cette résistance :

- la perte d'autorité sur l'architecture ;
- la conception incrémentale en remplacement de l'analyse préliminaire détaillée.

4.2 L'AUTORITÉ DE L'ARCHITECTE LOGICIEL

Principe agile : Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto-organisent.

L'un des principes agiles important est la création d'équipes pluridisciplinaires ayant les compétences nécessaires pour réussir le développement de solutions logicielles de manière autonome. Lorsqu'une équipe ne réunit pas les compétences requises à son mandat, l'organisation a la responsabilité d'entourer cette équipe avec des spécialistes.

La meilleure alternative est d'adjoindre à l'équipe les compétences qui la rendront autonome. Cependant les spécialistes, parce qu'ils sont souvent peu nombreux supportent plusieurs équipes. Leur disponibilité devient alors un enjeu pour une équipe qui dépend d'un membre externe pour réussir ses engagements.

Un architecte, qui appuie une équipe autonome mais ne peut s'engager avec elle doit prendre garde de ne pas devenir un goulot d'étranglement. C'est pourquoi il est invité à se positionner comme un conseiller plutôt que comme une figure d'autorité ou de conformité. Le défi des conseillers, c'est que bien qu'ils soient compétents, l'équipe n'a pas l'obligation d'écouter tous leurs conseils.

Comme l'explique Mike Cohn, un architecte qui n'a pas la reconnaissance de ses pairs risque de ne pas être sollicité. Un architecte ayant toujours eu le pouvoir décisionnel sur la conception des solutions, mais étant également contesté par les équipes de réalisation, risque de se faire écarter dans un contexte agile. Surtout si :

- Les membres des équipes ont collectivement les compétences requises[1].
- L'architecte a abusé de son autorité dans le passé, en imposant ses idées sans prendre en considération l'opinion des équipes de développement, ou en communiquant principalement par échange de documentation. Ce phénomène, où l'architecte spécifie le travail à faire sans s'impliquer dans la réalisation, est appelé le « syndrome de la tour d'ivoire ».

Un architecte avec une expertise reconnue est sollicité par ses pairs qui entendent profiter de son expérience. Surtout s'il a l'habitude de les consulter pour confirmer ses choix et qu'il prend la peine d'expliquer ses propositions suite à la sélection d'une solution.

Donc, l'architecte agile se doit d'être un bon communicateur, d'être collaboratif et de respecter l'expertise de ses pairs.

4.3 LA CONCEPTION INCRÉMENTALE COMPARÉE À L'ANALYSE PRÉLIMINAIRE DÉTAILLÉE

Plusieurs responsables d'architecture ont l'habitude de démarrer les projets avec une analyse préliminaire plutôt détaillée tandis que les approches agiles préconisent la

1. D'ailleurs Scott Ambler, précise bien que l'architecture logicielle n'est pas qu'une question d'élite et qu'il n'est pas nécessaire d'être un architecte pour concevoir celle-ci. Sur son site http://www.agilemodeling.com/, il propose de remplacer le titre d'architecte par un rôle de responsable d'architecture, à l'exemple du rôle de responsable de produit de la méthode Scrum. Nous allons suivre ce conseil et nous référerons aux responsables d'architecture plutôt qu'aux architectes à la suite de cette section.

conception incrémentale. À l'extrême, cette stratégie de l'agilité limite les travaux de la conception aux seules spécifications concernant l'engagement de l'itération suivante. Cette approche est motivée par le principe agile suivant :

> **Principe agile** : Un logiciel fonctionnel est la meilleure unité de mesure de la progression du projet.

Évidemment, un projet complexe induira une architecture plus complexe et il peut sembler aberrant de concevoir à la pièce un tel système. L'idée est donc de trouver le juste milieu entre une conception tenant compte de la complexité et une analyse trop poussée.

La suite de ce chapitre explique les concepts à considérer lorsqu'une architecture est conçue de façon incrémentale.

4.4 OBJECTIFS D'UNE ARCHITECTURE AGILE

Le site *agilearchitect.org*[1] propose cinq objectifs visés par une architecture agile. Les prochaines sections expliquent comment ces objectifs balisent le processus de conception.

4.4.1 Livrer une solution fonctionnelle

> **Principe agile :** Notre première priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels utiles.

La livraison en continu d'une solution fonctionnelle vise à limiter au maximum l'effet tunnel[2]. D'un point de vue de l'approche agile *Lean*, cette accumulation de travail, risquant de ne pas satisfaire totalement les attentes d'un client, est un gaspillage potentiel.

C'est pourquoi les approches agiles encouragent à ne pas traiter les disciplines de développement séquentiellement, mais à distribuer l'effort de chaque discipline à travers toutes les itérations, en considérant le produit de chaque itération comme une petite livraison finale.

> Réaction d'un détracteur :
> « *Notre projet construit la première version de la plateforme technique qui sera utilisée par notre entreprise dans les cinq prochaines années. On ne peut pas l'limiter les travaux d'architecture aux seules considérations de la prochaine itération. Nous avons besoin de temps préalable pour concevoir la bonne plateforme.* »

1. Source : Agile Architect, http://www.agilearchitect.org, consulté en octobre 2010.
2. L'effet tunnel est une analogie utilisée pour représenter les projets qui déçoivent les attentes des clients, provoqué par des disciplines enchaînées en cascades, retardant la démonstration des résultats d'une solution logicielle à la fin d'une phase ou du projet.

Le danger de limiter les travaux de conception au départ d'un projet est de livrer des solutions faciles, qui certes livrent rapidement beaucoup de fonctionnalités visibles pour les clients, mais ne répondent pas aux grands enjeux du projet, comme la performance ou la sécurité.

Pour éviter cet égarement, nous conseillons de faire des travaux de conception concentrés sur la vision d'architecture dès le démarrage du projet. De la même façon qu'une vision d'affaires aidera le client à planifier le développement de fonctionnalités d'un système, une vision d'architecture aidera les responsables d'architecture à ne pas perdre la vue d'ensemble du système à développer. La vision d'architecture est donc une cible plutôt qu'une certitude.

Scott Ambler identifie[1] quelques raisons de prendre du temps pour faire la conception de la solution technique avant le démarrage d'un projet :

- augmenter la productivité, en évitant de s'engager sur des alternatives ne répondant pas adéquatement aux besoins ;
- faire les preuves de concepts pour diminuer les risques techniques ;
- estimer les ressources nécessaires (budget et échéancier) ;
- planifier l'expansion du système ;
- mieux communiquer avec les membres impliqués ;
- permettre aux équipes de s'approprier l'architecture cible.

Cependant, lorsque Scott parle de conception préalable, il fait référence à une courte itération ne durant que quelques jours.

> Réaction d'un détracteur :
> « *Nous avons négocié avec le client que nos deux premières itérations serviront à construire l'architecture du système. Il pourra donc commencer sa planification à partir de la troisième itération.* »

En commençant le développement rapidement, les incréments de logiciel deviennent une façon de vérifier les hypothèses de conception et de mesurer l'état d'avancement du projet. Il est normal que les premières itérations comportent plus de travaux d'architecture, même si cela veut dire très peu de fonctionnalités à démontrer à la fin de l'itération.

Dans une approche agile, le client ne planifiera pas des itérations dans lesquelles l'équipe ne livre aucune fonctionnalité d'affaires. Il est alors toléré de réduire le nombre de fonctionnalités à développer au cours des premières itérations. La prochaine section illustre comment une équipe de réalisation conjugue à la fois livraison de fonctionnalités et développement technique.

1. Sur son site www.agilemodeling.com.

4.4.2 Maximiser la valeur d'affaires

L'approche agile *Scrum* prévoit que le responsable de produit soit responsable de la planification du projet. En considération de la portée du projet et en s'appuyant sur le contenu du carnet de produit il planifie les fonctionnalités qu'il souhaite voir développer d'une itération à l'autre.

Du point de vue d'un responsable affaire, il est difficile de percevoir la valeur des efforts techniques, simplement car c'est une abstraction visible uniquement des personnes initiées à la technique. Cela demande de la créativité et de la collaboration de la part des responsables d'architecture pour initier le responsable de produit à ne pas négliger la planification technique.

Une première étape est d'identifier la valeur d'affaires apportée par le développement technique. Par exemple, si le responsable de produit ne comprend pas la valeur d'affaires associée à la sécurité et la considère comme un aspect technique du système, il risque de ne jamais planifier son développement. Il considère sans doute que c'est un aspect implicite et sans efforts : personne ne voudrait permettre l'accès libre à un système lorsqu'il comporte des informations critiques.

Puisque protéger les informations sensibles d'un système est une préoccupation d'affaires, le carnet de produit devrait inclure un item rédigé par le responsable de produit et décrivant les besoins en sécurité aux responsables de l'architecture. L'exemple de la sécurité peut sembler évident, mais il existe d'autres sujets comme la performance (quels sont les temps-réponse attendus par le client ?) ou la journalisation (quel est le besoin d'affaires de garder la trace des transactions ? Qui consommera ces traces ?). Ce questionnement concrétise la valeur d'affaires autour des considérations techniques et donne un intérêt pour le client à les inclure dans sa planification.

Retour d'expérience

Dans le cadre d'un projet de développement de compte à haut rendement pour les clients d'une compagnie d'assurance, l'équipe avait décidé de développer l'affichage du solde au cours de la première itération :

« *En tant qu'épargnant, je voudrais pouvoir vérifier le solde de mon compte pour m'assurer d'avoir suffisamment épargné pour financer mon projet personnel.* »

La condition de succès de cette fonctionnalité aurait pu se limiter à vérifier l'affichage de la page du solde, et se limiter au contexte de la couche web du projet.

Mais la condition de succès était plutôt de s'assurer que le solde affiché était identique à celui présenté au guichet bancaire. Avec une telle condition de succès, la fonctionnalité permettait à l'équipe de réalisation de tester une première fois l'interaction de la couche web avec la base de données partagée, avec le système COBOL et avec les systèmes financiers de l'organisation.

La fonctionnalité avait à la fois une valeur d'affaires et une forte importance technique.

Évidemment, l'équipe aurait pu s'engager à développer d'autres fonctionnalités si la condition de succès avait seulement été de vérifier l'affichage correct du solde. Mais la condition de succès comparant le solde à celui du guichet bancaire était un bien meilleur indicateur de l'état d'avancement du projet.

> C'était un bon compromis entre livrer de la valeur d'affaires et concevoir la plateforme système.
> Mathieu

Chacune des fonctionnalités du carnet de produit devient une opportunité de concevoir le système et permet à l'équipe de réalisation de supporter le client dans la rédaction des spécifications d'affaires de manière à ce que chacune des livraisons donne l'état d'avancement du projet. Cette approche suppose que ce sont les besoins d'affaires qui justifient le reste de la conception détaillée et que la planification est un aspect de négociation entre les responsables de l'architecture et le client.

4.4.3 Prévoir les prochains travaux

La période de démarrage est probablement la période qui a le plus d'impact sur la suite du projet. En particulier lorsqu'il doit être assuré que les membres d'une équipe de développement de dix personnes seront en mesure de débuter la réalisation sans se nuire dès la première itération.

Si votre entreprise a l'habitude d'investir dans une conception détaillée de plusieurs semaines, voire plusieurs mois, pourquoi ne pas proposer une alternative plus agile ? Comme d'adjoindre un ou deux développeurs aux responsables de l'architecture pour à la fois réaliser itérativement une conception de haut niveau et vérifier les points d'architecture par des preuves de concept. Une telle équipe est également en mesure de livrer le chemin critique fonctionnel, tel que proposé dans la pratique du *Story Mapping* du chapitre *3 Démarrage d'un projet agile*. De cette manière, au terme de la période de démarrage, la livraison incrémentale sera engagée, les risques techniques seront mitigés et l'investissement dans le projet sera limité.

La situation sera idéale pour prendre une décision éclairée sur la poursuite du projet basé sur des résultats fonctionnels et techniques. C'est au moment où le client désire poursuivre le projet que la croissance de l'équipe devient pertinente : une recherche de productivité suite à des risques mitigés ! N'est-ce pas, au fond, le véritable objectif recherché par les adeptes de l'analyse préliminaire détaillée ?

Dans une approche incrémentale, les fonctionnalités développées au début du projet seront plus coûteuses à développer parce qu'elles nécessitent plus d'activités exploratoires pour être livrées. Plus tôt les spécifications seront construites pour s'assurer de la faisabilité technique du système, plus tôt le risque technique sera mitigé. Plus tôt le risque technique sera mitigé, plus tôt il y aura de place pour développer des fonctionnalités d'affaires.

Pendant la réalisation, il faut garder du temps pour poursuivre les activités de conception et les preuves de concept. Le carnet de produit étant continuellement raffiné cela retarde certaines décisions ayant un impact sur le coût de développement associé aux fonctionnalités.

Il est primordial que l'équipe modifie certaines pratiques de développement afin d'être en mesure d'intégrer les nouveaux choix de conception qui surviendront en cours de projet. Par exemple :

- Concevoir une architecture modulaire faiblement couplée permet de limiter l'effort nécessaire lorsque survient le besoin de remplacer des implantations.
- Avoir un plan de tests unitaires qui couvre une grande partie du code permet de modifier en toute confiance du code existant en réduisant le risque d'introduire de la régression.

Si certains croient que la gestion du changement ne s'applique pas pour eux car elle ne concerne que la gestion des opportunités, nous leur rappelons qu'elle concerne également les correctifs et qu'il est primordial d'être préparé à cette éventualité. Ces pratiques sont principalement abordées par la méthode *eXtreme Programming* qui accorde une grande importance au remaniement de code.

Le livre *Working Effectively with Legacy Code* de Michael C. Feathers est également une bonne référence pour une équipe de réalisation qui voudrait être habile pour modifier du code existant, autant dans le système développé que dans un système patrimonial.

4.4.4 Répondre à des besoins d'affaires réels

Il arrive que le manque de compétences, l'obligation de se conformer aux orientations corporatives, la tentation d'essayer de nouvelles technologies ou la résolution de produire du code plus élégant, amènent les équipes à proposer des solutions techniques complexes. Ces choix techniques amènent parfois les équipes à négocier les besoins du client pour des questions de faisabilité. Une équipe doit se remettre en question lorsqu'elle se surprend à argumenter dans le seul but de justifier les choix technologiques, s'éloignant ainsi du développement d'une solution utile.

Souvent en cours d'itération, les développeurs sont tentés de terminer un module ou une fonctionnalité, même si ce travail supplémentaire n'est pas planifié ou démontrable. Bien qu'issu d'une bonne intention, cela introduit une dette technique et constitue une autre source d'éloignement à la livraison d'une solution utile et vérifiable.

4.4.5 Gérer le changement

Fixer une cible d'architecture qui sera validée à travers les fonctionnalités à développer limite considérablement le changement qui pourrait survenir pendant le projet. Cependant, cela n'élimine pas la possibilité qu'une nouvelle opportunité devienne la plus grande priorité du moment. Et malgré toutes les précautions, certaines problématiques critiques resteront toujours imprévisibles. Nous ne sommes donc jamais à l'abri de devoir changer la portée initiale d'un projet. C'est d'ailleurs l'un des principes qu'honorent les approches agiles :

> **Principe agile :** Le changement est accepté, même tardivement dans le développement. Les processus agiles exploitent le changement comme avantage compétitif pour le client.

Comme c'est une des hypothèses de travail entendue entre l'équipe de développement et le client, il faut être prêt à recevoir des demandes de changements.

C'est un point très attrayant pour les clients qui ont vécu la difficile expérience des demandes de changement. Dans un contexte de gestion de projet en cascade, les demandes de changement sont onéreuses parce qu'elles impliquent de revoir l'analyse, de modifier la conception, de revoir du code, etc. C'est un imprévu de coût et de temps qui peut mettre en péril la bonne tenue d'un projet.

Avec une approche itérative, le client peut introduire à chaque itération de nouvelles demandes, ce qui réduit l'importance des demandes de changement. Il faut cependant être conscient de l'impact qu'a cette approche sur la gestion du projet. En premier lieu, c'est coûteux ! Notre collègue, François Beauregard de Pyxis Technologies, avait l'habitude de dire :

> « *On peut choisir la couleur de sa tuile de céramique autant de fois que l'on veut. Mais une fois installée, c´est plus coûteux !* »

Pour éviter de reprendre des fonctionnalités déjà livrées, il faut s'habituer à les planifier au dernier moment responsable possible. C'est-à-dire que tant qu'une fonctionnalité est imprécise, et tant qu'elle n'a pas d'impact sur le bon déroulement du projet, le client devrait patienter avant de la planifier. Le site *agilearchitect.org* rappelle cette maxime à propos des demandes de changement :

« *Don't fight it, embrace it, but plan for it.* »[1]

Cette maxime résume bien l'état d'esprit dans lequel le changement devrait être accueilli.

Un premier outil qui peut aider à l'éducation du client, c'est une charte de projet, courte mais claire. Avec une charte de projet, il est possible de challenger une nouvelle demande pour vérifier qu'elle répond aux objectifs du projet. Cette évaluation permettra de prioriser cette demande parmi celles initialement prévues.

4.5 DOCUMENTATION

Même si cet ouvrage comporte un chapitre sur la documentation agile, cette section présente quelques conseils pour la documentation de l'architecture, un domaine où la documentation est souvent un point sensible.

1. N'oubliez pas que la communication entre les responsables de l'architecture et les équipes de réalisation est plus importante que la documentation. Évitez la communication par échange de documents qui entraîne le syndrome de la tour d'ivoire.

1. Ce qui se traduit par « Ne les combattez pas, étreignez-les, mais planifiez-les. »

2. Documentez au dernier moment responsable. Comme les fonctionnalités qui sont détaillées au fur et à mesure, documentez aussi votre architecture de manière incrémentale.
3. Pour éviter de documenter des concepts inutilement au nom de la pérennité, cherchez à savoir qui en sont les consommateurs et comment ils utiliseront votre documentation. Si la documentation est temporaire et n'est utile qu'à la communication, privilégiez d'autres formes de documentation, moins formelles. Par exemple, la photo d'un tableau blanc est peut-être suffisante.
4. La documentation est un livrable. Pour votre organisation, si la documentation de l'architecture est essentielle pour la pérennité du système, alors cette documentation devrait être construite de façon incrémentale, et validée de façon itérative, comme les incréments de logiciel.
5. Parce que la documentation est un livrable à construire de manière incrémentale, cherchez une structure qui facilite le changement régulier. Si possible, profitez des outils électroniques qui peuvent vous faciliter la tâche. Par exemple, il est rarement utile de maintenir manuellement un diagramme de classes lorsqu'il est possible de le générer automatiquement.

Pour une description plus complète et générale, nous vous invitons à lire le chapitre *14 Agilité et documentation*.

En résumé

Le développement incrémental modifie la stratégie de conception de l'architecture. L'analyse détaillée, empreinte de la bonne intention de mitiger les risques d'un projet, peut échouer parce que :

– l'analyse ne démontre pas l'état d'avancement du projet ;

– les documents ne sont pas à l'abri des erreurs ;

– l'analyse détaillée peut être longue, ce qui limite la réaction au changement.

Plutôt que de considérer la conception comme une activité préalable au projet, il faut chercher à la distribuer tout au long du projet. Pour y parvenir, il faudra adopter certains objectifs :

– Livrer une solution fonctionnelle : se servir des fonctionnalités à développer comme moyen d'introduire et démontrer la réalisation technique ;

– Maximiser la valeur d'affaires : de concert avec le client, planifier à la fois la livraison de fonctionnalités et la construction du système ;

– Prévoir les prochains travaux : être agile, ce n'est pas jouer à l'autruche. Il est possible de concevoir juste suffisamment à l'avance pour éviter les problèmes, tout en limitant l'effort pour éviter le gaspillage potentiel ;

– Répondre aux besoins d'affaires : les choix technologiques ne doivent pas restreindre le développement orienté par les besoins d'affaires ;

– **Gérer le changement :** les changements sont les bienvenus, alors planifions la solution logicielle en conséquence.

Le rôle de l'architecte peut être amené à changer, surtout s'il est impossible pour lui d'être engagé au sein des équipes de réalisation. Les architectes qui auront le plus de succès à intervenir dans les approches agiles sont de bons communicateurs, ils sont collaboratifs et ils respectent l'expertise de leurs pairs et vice-versa.

L'échange de documents n'est pas un moyen efficace de diffuser les choix d'architecture. La documentation est cependant importante pour assurer la pérennité du système. C'est pourquoi il faut la considérer comme un des livrables du système et la produire de façon incrémentale, au même rythme que la solution logicielle.

5

Équipes et livraison incrémentale

Objectif

Dans ce chapitre, vous découvrirez en quoi consiste la livraison régulière d'un incrément de logiciel fonctionnel. Vous comprendrez pourquoi ce mode de travail requiert d'établir une définition de « terminé » au regard de la gestion de la « dette technique ». Vous découvrirez la façon de constituer une équipe de projet autonome et multidisciplinaire. Vous serez en mesure d'identifier les répercussions possibles de cette composition d'équipe sur différents corps de métier du développement logiciel. Finalement, vous constaterez les impacts positifs de l'amélioration continue.

Mise en situation : Claude et le projet bancaire sur internet

L'équipe Alpha développe une application web pour offrir la gestion des comptes bancaires personnels aux clients de leur institution bancaire. À terme, les clients seront en mesure d'ouvrir eux-mêmes différents comptes bancaires. L'architecture a prévu une application web qui devra s'interfacer avec le système central en COBOL pour échanger des données avec les autres services de la banque.

L'équipe est prête à démarrer le projet et elle commence par vérifier le travail qui pourra être accompli pendant la première itération. La première fonctionnalité envisagée est celle de l'affichage du solde avec l'hypothèse que le client n'a qu'un seul compte. L'affichage de plusieurs comptes fera l'objet d'une autre spécification, quand le client sera en mesure de créer des comptes lui-même.

Claude, le client : Combien d'autres spécifications l'équipe de projet sera-t-elle en mesure de prendre pour cette première itération ?

Équipe : Cela dépend de tes attentes face à la réalisation de la fonctionnalité. De notre côté, nous avons une définition de terminé normalisée que nous nous engageons à respecter pour assurer une qualité uniforme pour toutes les fonctionnalités livrées. Cette définition comprend le développement d'une série de tests unitaires, la revue du code, un incrément de logiciel déployé sur un environnement de tests d'acceptation que tu pourras utiliser pour faire des démonstrations.

De ton côté, nous savons déjà que tu ne t'attends pas à voir plus d'un solde dans le bilan du compte client. Mais pour être sûr d'avoir bien compris l'ampleur du travail que tu nous demandes, peux-tu nous expliquer les tests que tu exécuteras pour vérifier que nous avons correctement répondu à tes exigences ?

Claude : Et bien, je vais me connecter sur mon propre compte, puisque je n'ai qu'un seul compte bancaire, et je vais vérifier que le compte correspond bien à celui que je peux voir sur un guichet automatique[1].

Équipe : D'accord. Cela veut dire que nous devrons traverser la couche web et la couche COBOL, et qu'il faudra prévoir un échange de fichiers avec le système financier. Nous serions alors en mesure d'accéder aux données de pré-production. Est-ce suffisant pour le moment ?

Claude : Bien sûr. Alors sur combien d'autres spécifications êtes-vous capables de vous engager ?

Équipe : Aucune. Cette fonction est très complexe mais elle a beaucoup de valeur. Elle nous permettra de vérifier que notre système s'interface bien avec les autres systèmes en place. C'est un bon test pour notre architecture. Est-ce que nous sommes d'accord pour dire que l'interface graphique aura un air de prototype mais que la fonctionnalité sera bel et bien opérationnelle ?

Claude : Oui, c'est même mieux. Je n'étais pas encore prêt à vous fournir la maquette graphique. Un graphiste travaille toujours sur sa conception.

Équipe : Entendu. On s'en reparle à une prochaine itération. A priori l'engagement est raisonnable et nous aurons terminé dans un mois. Nous allons tout de même contre-vérifier en découpant le scénario en tâches. Si le découpage soulève un imprévu, nous en discuterons avec toi.

Claude : Parfait, je vais détailler de nouvelles spécifications en attendant la démonstration du résultat.

Principe agile : Livrer fréquemment une application fonctionnelle, toutes les deux semaines à deux mois, avec une tendance pour la période la plus courte.

Le principe de la livraison incrémentale implique que chacune des itérations est un petit projet dont l'objectif est l'enrichissement graduel de la solution globale. Cette façon de conduire les projets de développement transforme certaines activités qui ne peuvent plus être tenues de manière séquentielle, comme la planification, la réalisation, la livraison et l'introspection.

Voyons comment chacune de ces activités peut être distribuée à l'intérieur des itérations, pour s'adapter à la réalité du développement incrémental.

1. Terminal bancaire ou distributeur de billets de banque.

5.1 LA PLANIFICATION

5.1.1 La planification itérative

Les praticiens agiles planifient presque tous à l'aide des itérations, qui sont des périodes de temps limitées. À chacune des itérations, le client et l'équipe de projet se mettent d'accord sur le travail qui sera exécuté : le client se base sur les priorités de son carnet de produit et l'équipe de projet se base sur sa capacité à livrer du logiciel pendant la période.

Au terme de chacune des itérations, l'état d'avancement du projet est mesuré et une nouvelle itération est planifiée au regard de la dernière version du logiciel livré.

La planification devenant une activité régulière, il n'est plus nécessaire de planifier l'exécution dans sa totalité au commencement du projet. Le côté négatif de la chose est que cela oblige le client à se dédier à la gestion du projet. Le côté positif est que cette stratégie offre d'intéressants leviers d'ajustement comme :

- corriger les spécifications du carnet de produit en réaction au dernier incrément de logiciel livré ;
- ajouter et planifier de nouveaux éléments dans le carnet de produit, au gré des opportunités se présentant en cours de projet.

5.1.2 La longueur des itérations

Un billet de Tremeur Balbous[1] explique bien comment les itérations peuvent être utilisées comme des points de visibilité. La longueur des itérations déterminera le nombre de points de visibilité qui seront disponibles pendant la durée d'un projet. Avec une longueur de quatre semaines, un projet de six mois a environ cinq points de visibilité pendant son déroulement. Si le projet est complexe et risqué, cela n'est peut-être pas suffisant. Pour en ajouter, la réduction du temps des itérations peut être envisagée.

Un autre point à considérer est la capacité de l'équipe de projet à réaliser des exigences. Lorsqu'une équipe s'engage à réaliser des fonctionnalités, elle s'engage à livrer un incrément de logiciel de qualité « production », que le client est en mesure de vérifier lui-même. Des itérations d'une semaine offrent beaucoup de points de visibilité, mais demandent aussi de la maturité de la part de l'équipe de projet. Celle-ci devra être en mesure de produire le code, de le vérifier, de le déployer et de le démontrer. Toutes ces activités risquent de créer de la contingence et réduire le temps effectif disponible pour réaliser des exigences.

D'ordinaire, la durée des itérations n'excède pas six semaines et il est fréquent qu'elles durent entre deux et quatre semaines. Plusieurs équipes de projet considèrent que les intervalles de moins de deux semaines imposent un rythme insoutenable.

1. Tremeur Balbous, http://www.agilegardener.com/2010/04/22/les-points-de-visibilite-en-scrum-comment-choisir-la-longueur-des-iterations/, consulté en octobre 2010.

> Réaction d'un détracteur : « *Je ne vois pas comment notre équipe serait en mesure de livrer un incrément de logiciel fonctionnel en seulement 3 semaines.* »

Une équipe ayant besoin d'itérations plus longues travaille probablement avec des langages ou du code ne permettant pas les tests automatisés et dont les composantes logicielles comportent de fortes dépendances. Dans un tel contexte, nous proposons d'allonger la durée des itérations, mais en se fixant comme objectif de réduire la durée le plus tôt possible, en réfléchissant ensemble sur les obstacles empêchant aujourd'hui l'adoption d'itérations plus courtes.

5.1.3 Le carnet d'itération

Connaissant la capacité de l'équipe, le carnet de produit ordonné et une durée d'itération entendue, le client et l'équipe de projet sont en mesure de négocier les spécifications à livrer au terme de la prochaine itération. Les spécifications tirées du carnet de produit devraient être constituées de manière à ajouter de nouvelles fonctionnalités au système. Par contre, elles seront peut-être trop grandes pour permettre le suivi quotidien de la santé de l'itération. C'est pourquoi les spécifications sont généralement découpées en tâches. Ce découpage fin dresse la liste de ce qui doit être fait afin de livrer le prochain incrément de logiciel.

Comme décrit dans l'ouvrage *Scrum : Le guide pratique de la méthode agile la plus populaire* de Claude Aubry, il existe deux types de tâches :

- Les tâches déduites des exigences qui découpent avec une granularité plus fine le travail à réaliser pour livrer les fonctionnalités.
- Les tâches indépendantes, c'est-à-dire toutes les tâches qui ne sont pas associées directement à une spécification mais qui sont à considérer dans le cadre de l'itération. Par exemple, les rencontres exceptionnelles, les tâches récurrentes incluses dans la définition de terminé, la résolution d'un obstacle, une action décidée lors de la rétrospective précédente, ou une activité pour améliorer la qualité du produit.

Cette liste de tâches sert à l'équipe pour organiser quotidiennement le suivi de l'itération :

- En additionnant l'effort de l'ensemble des tâches de l'itération, l'équipe peut vérifier que l'engagement pris avec le client est réaliste. Dans la négative, elle peut renégocier la portée de l'itération avant la réalisation.
- Avec un tableau des tâches, l'équipe peut planifier la réalisation de l'itération. Ce tableau informe quotidiennement de l'état de chacune des tâches, de l'ensemble du travail restant à l'itération, de l'occupation et des obstacles de chacun des membres de l'équipe.
- Avec un graphique d'avancement, l'équipe peut surveiller la tendance générale de l'itération. À partir de cette tendance, elle peut adapter sa stratégie de réalisation, s'assurant que leur engagement ne leur échappe pas.

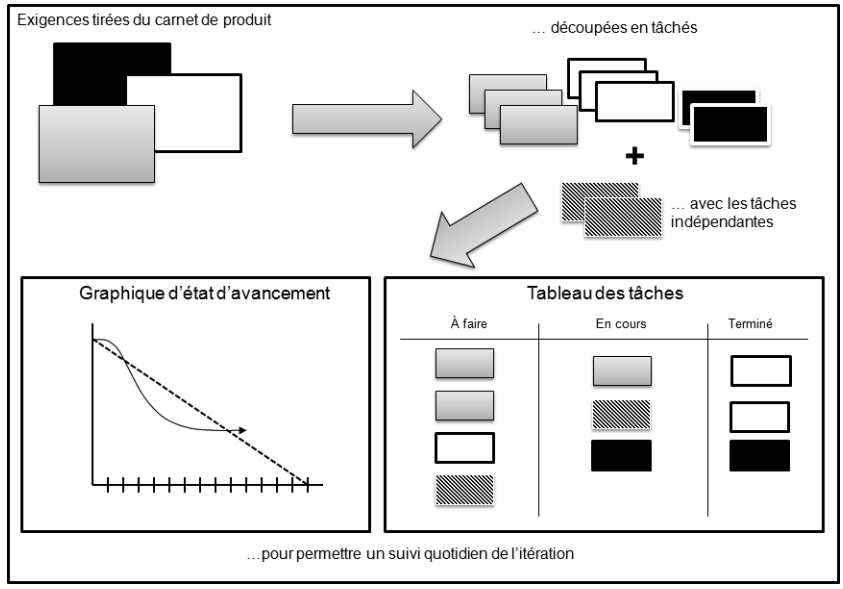


Figure 5.1 — Le carnet d'itération est un outil facilitant la planification et le suivi des itérations.

Pour faire un suivi quotidien, nous conseillons de découper les spécifications en tâches fines. Un conseil généralement reconnu par les praticiens agiles est de limiter la taille des tâches inférieure à deux jours de travail, tout en essayant de viser des durées plus courtes.

5.1.4 La planification des travaux techniques

Les spécifications composant le carnet de produit décrivent souvent des fonctionnalités plutôt que des travaux techniques. Certaines équipes de projet se préoccupent de ne pas les retrouver à l'intérieur du carnet de produit. Ils sont inquiets que certains aspects techniques importants soient inconnus du client et qu'ils ne soient donc jamais planifiés.

D'ordinaire, le client comprend que les travaux techniques sont nécessaires pour réaliser des fonctionnalités. Ce qui est parfois difficile à comprendre, c'est que l'effort de ces travaux n'est pas constant selon l'itération dans laquelle une fonctionnalité est planifiée (voir Figure 5.2).

Cette variation s'explique par le fait que certains travaux techniques sont introduits par une spécification, mais sont potentiellement utiles à plusieurs autres. Par exemple, si une première fonctionnalité nécessite un contrôle d'accès protégé, il est envisageable que le module de sécurité qui sera développé en conséquence sera utile pour plusieurs autres fonctionnalités.

Avec l'hypothèse que les concepts techniques sont toujours introduits par une demande fonctionnelle, le besoin d'introduire des travaux techniques dans le carnet

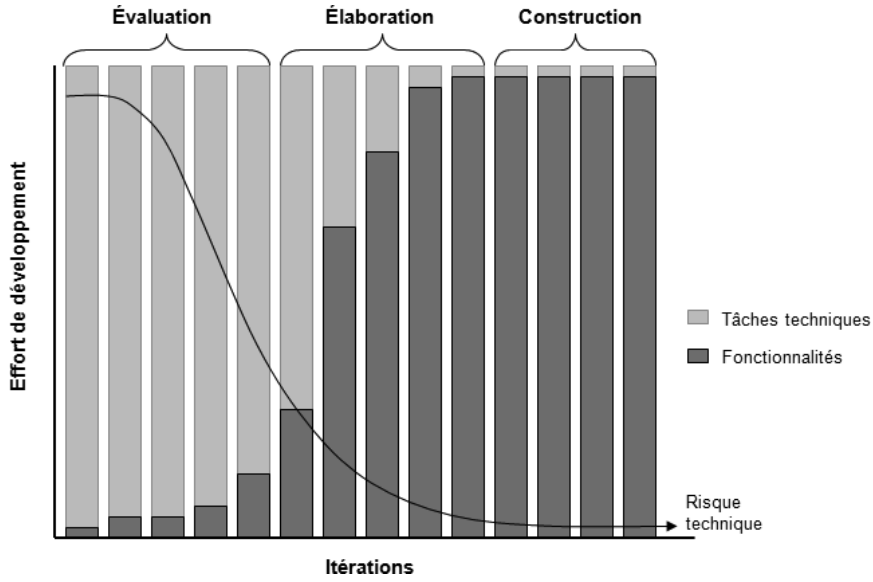


Figure 5.2 — Productivité du développement de fonctionnalités versus l'effort requis des travaux techniques.

de produit est moins grand. Dans ce cas, le client accepte que la proportion de travaux techniques diminue avec le temps, ce qui donne une impression de vitesse grandissante de développement de fonctionnalités pendant le déroulement du projet.

Il est parfois difficile d'introduire un aspect technique à travers l'ajout d'une fonctionnalité. Dans ce cas, rien n'empêche d'ajouter une tâche technique au carnet de produit, pour qu'elle n'échappe pas au client et que sa planification ne soit pas oubliée.

5.1.5 L'entretien du carnet de produit

Comme les approches agiles préconisent d'éviter l'analyse détaillée préliminaire, l'entretien du carnet de produit doit d'être distribué tout au long du projet. Le rythme de l'analyse doit être suffisant pour soutenir la cadence de réalisation des équipes de projet. De plus, le résultat de cette analyse doit être suffisamment précis pour permettre à l'équipe de projet de s'engager de manière fiable, par exemple en produisant des spécifications aux qualités INVEST[1]. Les activités d'analyse, parallèles à celle des itérations de développement, sont une charge de travail exigeante qui occupe une grande proportion du temps du client.

Pour un rythme soutenu et soutenable, nous conseillons de conserver en tout temps l'équivalent de deux itérations de spécifications prêtes à réaliser.

1. Rappel de l'acronyme INVEST : **I**ndépendant, **N**égociable, avec une **V**aleur d'affaires, **E**stimable, **S***mall* (petite) et **T**estable.

5.2 LES ITÉRATIONS

> **Principe agile :** Notre première priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels utiles.

La livraison régulière de logiciel est un concept parfois difficile à saisir. Qu'est-ce qui est sous-entendu par les termes « livrer » et « régulièrement » ?

« Livrer » sous-entend la réalisation complète d'un incrément de logiciel aussi opérationnel que s'il était livré dans une version finale. Cependant, cela ne veut pas forcément dire que chacun des incréments comprend suffisamment de fonctionnalités pour être distribué aux utilisateurs finaux. Cette condition pourra demander plusieurs itérations avant d'être atteinte.

« Régulièrement » signifie à une cadence régulière, à un rythme soutenable.

> Réaction d'un détracteur : « *Attendez ! Je ne suis pas sûre de comprendre. Je ne vois pas comment il est possible de livrer du code de qualité production à chacune des itérations. Les spécifications sont beaucoup trop grandes pour être contenues dans une seule itération. De plus, il faut prévoir du temps de conception. C'est sans compter le temps nécessaire pour déployer la solution.* »

Au risque de paraître utopiques et de vous voir fermer ce livre avec un soupir de découragement, nous confirmons que vous avez bien compris. Lorsque nous parlons de livrer du logiciel de manière itérative, nous voulons dire faire toutes les activités nécessaires pour déployer une nouvelle version du produit. C'est possible puisque bon nombre des équipes que nous avons accompagnées y sont parvenues.

5.2.1 La revue d'itération

La revue d'itération, parfois vulgarisée par « démonstration », est une pratique qui consiste à présenter chacun des incréments au client. C'est un moment d'accomplissement où l'équipe affiche le résultat des travaux de l'itération et se synchronise avec le client sur l'état actuel du produit.

À la fin de l'itération, le client est responsable d'accepter ou de refuser le résultat obtenu. Cela implique qu'il est primordial de s'entendre sur les résultats attendus. Dans son livre *User Stories Applied : For Agile Software Development*, Mike Cohn explique qu'une spécification (sous forme de *user story*) devrait avoir une condition de succès qui fixe sa portée et réduit les risques d'avoir de fausses attentes au moment de la revue de l'itération.

Pour une équipe s'initiant à une approche agile, il n'est pas rare qu'à la première itération, les conditions de succès initialement établies n'aient pas permis de se comprendre entièrement. Il est donc essentiel de voir la revue d'itération comme une opportunité pour revoir et améliorer la forme et le détail des spécifications.

Lorsque la mise en œuvre d'une spécification n'est pas terminée au moment de la revue d'itération, soit par manque de temps, soit parce que les conditions de

succès n'ont pas été rencontrées, il est de la responsabilité du client de refuser la fonctionnalité livrée. Il ne faut pas oublier que d'avoir des fonctionnalités inachevées empêche la livraison de l'application en production. C'est pourquoi l'équipe ne devrait pas présenter les fonctionnalités non terminées. Ceci doit également conduire l'équipe à reconsidérer son engagement lors de la planification de l'itération suivante.

Nous venons d'établir que le client est en mesure de vérifier ce qu'il peut observer : les fonctionnalités qui enrichissent le produit. Il reste cependant des éléments qui lui sont invisibles : la qualité interne et le travail restant pour que le produit puisse être livré en production. Un outil permet de rendre ces éléments visibles aux yeux du client : la définition de terminé.

5.2.2 La définition de terminé

La définition de « terminé »[1] est une description sans équivoque décrivant tout ce qui doit être fait pour considérer un item du carnet de produit comme étant complété. Cette définition doit être respectée à chacune des itérations. De plus, elle devrait être la plus exhaustive possible, en reprenant toutes les activités nécessaires pour que le produit soit livré en production.

> Réaction d'un détracteur : « *Notre définition de terminé comprend la revue par le groupe de référence. Allons-nous vraiment tester l'application avec notre groupe de référence à toutes les itérations ?* »

La réponse théorique est oui ! Autrement, vous acceptez d'accumuler une « dette technique » qui devra être remboursée avant de livrer votre application en production.

5.2.3 La dette technique

La dette technique est une analogie illustrant le travail s'accumulant de façon insidieuse pendant le développement d'un produit. Un billet[2] de Martin Fowler explique de la dette technique utilisée par les praticiens des approches agiles avec une métaphore de la dette financière. À l'image de la dette financière, la dette technique est un compromis sur la qualité dont la non-conformité s'accumule et devra être payée tôt ou tard avant la mise en production. Souvent, la décision d'accumuler une dette technique est prise pour atteindre une date de livraison importante, avec la ferme intention de rembourser cette dette une fois la livraison effectuée. Malheureusement, étant donné le rythme des projets dans les organisations, il est rare qu'une équipe ait la chance de reprendre les activités escamotées (par exemple : nettoyer son code au terme d'une livraison, compléter la documentation d'entretien, etc.).

Lorsqu'une équipe est mise sous la pression pour livrer dans des délais serrés, elle réussit souvent en produisant du code de moins bonne qualité, qui est plus facile à

1. Se référer au *3 Démarrage de projet agile* pour plus de détails sur la définition de terminé.

2. Martin Fowler, http://www.martinfowler.com/bliki/TechnicalDebt.html, consulté en octobre 2010.

écrire sur le moment mais qui demande l'utilisation de dépendances. Le risque de ne pas rembourser la dette est de créer puis conserver un code avec tant de correctifs temporaires, qu'il devient de plus en plus difficile, voire impossible de le modifier.

L'adhésion à une définition claire de terminé est une bonne façon de réduire l'accumulation d'une dette. Elle assure que l'engagement de l'équipe comprend tout ce qui est nécessaire pour livrer le produit en production. Si cette pratique ne devient pas une habitude, la capacité de l'équipe diminuera d'itération en itérations.

Une définition de terminé ciblant une qualité de production diminue les risques de rencontrer de mauvaises surprises au moment de la mise en production. Une mise en production implique souvent des aspects et des activités insoupçonnés pendant les activités de développement. Le lancement d'une mise en production en cours de projet, même partielle, permet de découvrir les aspects manquants à définition de terminé.

En conclusion, prenez garde à ne pas négliger les éléments de la définition de terminé. Cela donne peut-être l'apparence de livrer plus vite, mais c'est surtout prendre le risque de ne jamais rembourser la dette et de découvrir trop tard des lacunes engendrant un système plus coûteux à entretenir.

5.2.4 La stratégie de tests

Évidemment, la définition de terminé comprend son lot de tests. Ceux-ci sont révélateurs du changement de dynamique introduit par l'adoption des pratiques agiles. Janet Gregory et Lisa Crispin dans leur livre *Agile Testing: A practical Guide For Testers and Agile Teams*, présentent les différents tests logiciels répartis en quatre quadrants (voir Figure 5.3).

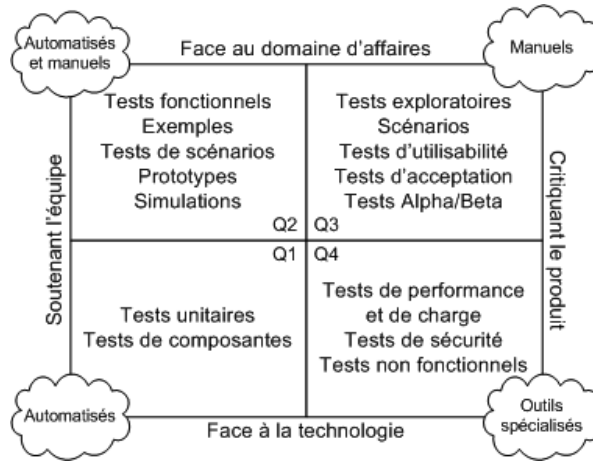


Figure 5.3 — Classification des types de tests logiciels.

Ces quadrants illustrent qu'il peut y avoir une grande quantité de tests nécessaires pour livrer un système en production. Est-il vraiment utile d'inclure tous les types de

tests à chacune des itérations ? Oui, lorsque le logiciel fonctionnel est utilisé comme mesure de l'état d'avancement des projets. Pour y parvenir, votre organisation doit être prête à exécuter les disciplines du développement logiciel en parallèle plutôt qu'en séquence, et ce peu importe la longueur de vos itérations.

Si les tests fonctionnels sont réalisés manuellement par des analystes, suite au travail des développeurs, nous pouvons prédire ceci : les analystes manqueront toujours de temps pour les compléter et la définition de terminé ne sera jamais respectée et risque d'être remise en question.

Une solution évidente à ce problème est de terminer le développement avant la fin de l'itération pour laisser aux testeurs le temps d'exécuter leurs plans de tests. Ce qui a pour conséquence d'arrêter le travail des développeurs qui se trouveront en attente pendant l'exécution des tests.

Une alternative est d'écrire le plan de tests avant la fin de la réalisation des spécifications. Les développeurs exécutent alors les tests eux-mêmes à la fin de la réalisation de la spécification pour s'assurer de sa conformité, réduisant ainsi la dépendance envers un testeur et augmentant le temps disponible de ce dernier.

En allant plus loin, lorsque les tests sont entendus entre les testeurs et les développeurs AVANT la réalisation, ces derniers sont en mesure de concevoir la solution autour des tests.

En poussant l'idée encore plus loin, lorsque les tests sont automatisés, les développeurs utilisent ceux-ci comme condition d'arrêt au développement : le travail est terminé seulement lorsque le produit se conforme aux tests. Cette façon de travailler s'inspire du cycle du *Test Driven Development*[1] (TDD) (Figure 5.4). Cette pratique de la méthode XP suggère de ne jamais écrire de ligne de code avant d'avoir écrit un test correspondant.

Évidemment, cela demande un changement dans les façons de faire, possiblement difficile, surtout dans les cas où les développeurs refusent de faire des tests ou que les testeurs ne veulent pas programmer des tests automatisés.

5.3 L'INTROSPECTION ET L'AMÉLIORATION CONTINUE

> **Principe agile :** À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accorde et ajuste son comportement dans ce sens.

Comme pour la planification, la réalisation et la livraison, l'introspection est une autre activité distribuée tout au long du projet. Nous avons vu que la planification n'est pas une activité rigide et qu'elle s'ajuste selon les résultats obtenus au terme des itérations. D'autres pratiques permettent également de faire de l'introspection en continu et cela à différentes fréquences. En voici quelques exemples :

1. TDD : développement piloté par les tests.

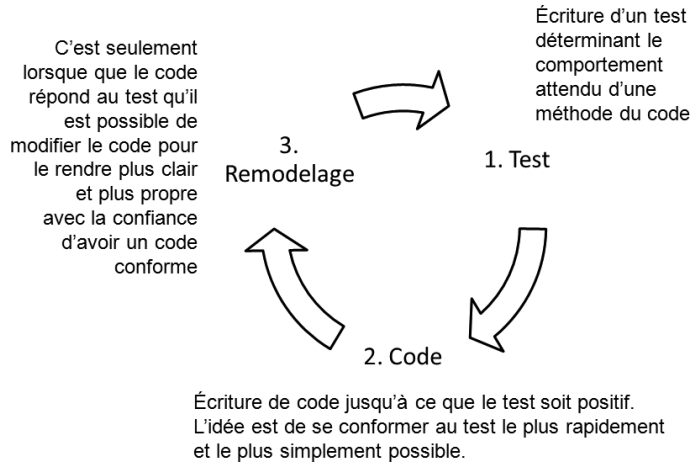


Figure 5.4 — Cycle du *Test Driven Development* (TDD).

- **La rétrospective** est la plus évidente puisque c'est un moment dédié à la remise en question. Elle revient à chaque fin d'itération. Pendant cette rencontre, l'équipe révise et questionne ses façons de faire et se dote d'un plan d'action pour s'améliorer.
- **L'intégration continue** consiste à exécuter le plan de tests unitaires automatisés à chacune des modifications de code sur le projet. Le plan de test devient un filet de sécurité couplé à un système de notification alertant les développeurs lorsqu'une régression est introduite.
- **La mêlée quotidienne** permet la planification de l'équipe pour la journée en cours. La santé de l'itération est ainsi vérifiée de façon quotidienne en observant le carnet d'itération.

Les rétrospectives sont parfois considérées comme une perte de temps et certaines organisations n'en voient pas l'utilité. Il existe des techniques pour s'assurer que ces rencontres demeurent efficaces. L'ouvrage *Agile Retrospectives : Making Good Teams Great*[1] est une bonne référence pour garder les rétrospectives pertinentes et efficaces. Ce livre propose un déroulement en cinq étapes :

1. **la mise en contexte** : rappeler les objectifs de la rencontre et le mode de fonctionnement de l'équipe ;
2. **le recueil d'informations** : réunir les faits qui se sont produits durant la dernière itération – les bonnes pratiques d'équipe à répéter et celles à modifier ;
3. **l'analyse** : découvrir les raisons expliquant le déroulement de l'itération et identifier celles qui sont efficaces et celles à améliorer ;

1. Esther Derby, Diana Larsen et Ken Schwaber, *Agile Retrospectives: Making Good Teams Great*, Raleigh : Pradmatic Bookshelf, 2006.

4. **le plan d'action** : rédiger un plan d'action SMART[1] adopté par toute l'équipe qui vise à améliorer le fonctionnement de l'équipe ;
5. **la conclusion** : évaluer si la rétrospective a atteint ses objectifs et si la formule mérite d'être revue.

L'objectif principal de la rétrospective est de construire et adopter un plan d'action. Pour atteindre ce but et éviter l'éparpillement, les participants doivent être attentifs à certains points :

- Éviter de cheminer sur des aspects qui n'aboutiront pas en un plan d'action. Une équipe ne maîtrise pas nécessairement toutes les causes de ses problèmes. Il est inutile de consommer tout le temps disponible à analyser des problèmes pour finalement se rendre compte qu'aucun plan d'action n'est possible. La solution dans de tels cas est d'avoir un plan d'action sollicitant la contribution du bon responsable.
- S'assurer d'avoir construit un plan d'action autour des améliorations les plus importantes et critiques. La gestion du temps disponible à l'analyse est importante ici, car il est facile dans le cadre d'une rétrospective de prendre trop de temps à analyser et de ne plus avoir assez de temps pour identifier un plan d'action pour tous les éléments discutés, ce qui devient frustrant pour les participants. Se concentrer sur quelques points importants est plus efficace et résulte en un meilleur plan d'action.

Pour éviter le gaspillage de temps, il est conseillé de contenir la rencontre dans une boîte de temps. La durée des rétrospectives dépendra de la longueur des itérations : des itérations de six semaines profitent d'avoir des rétrospectives plus longues que les itérations d'une semaine. Il est conseillé de faire une courte rétrospective à la fin de la rétrospective. C'est le bon endroit pour ajuster la longueur des séances et revoir la formule pour éviter que l'équipe ne se lasse de faire les mêmes exercices d'introspection. L'ouvrage *Agile Retrospectives : Making Good Teams Great* propose des ateliers différents pour prévenir la monotonie et adapter les rétrospectives aux besoins des équipes.

5.4 LA MISE EN PLACE DES ÉQUIPES MATURES ET AUTOGÉRÉES

Bien qu'il soit centré autour des ordinateurs et des langages de programmation, le développement logiciel est avant tout un processus humain : les bonnes équipes font de bons logiciels. Katzenbach et Smith, dans leur ouvrage *The Wisdom of Teams*, ont raffiné ce concept en illustrant quatre stades d'évolution d'une équipe. Olivier Devillard a évolué ce concept en cinq stades, avec lesquels nous faisons un parallèle

1. SMART : **s**pécifique, **m**esurable, **a**tteignable, pe**r**tinent (***r****elevant*) et limité dans le **t**emps.

avec les styles de leadership à adopter de façon à favoriser l'évolution de la cohésion des équipes[1] (voir Figure 5.5).

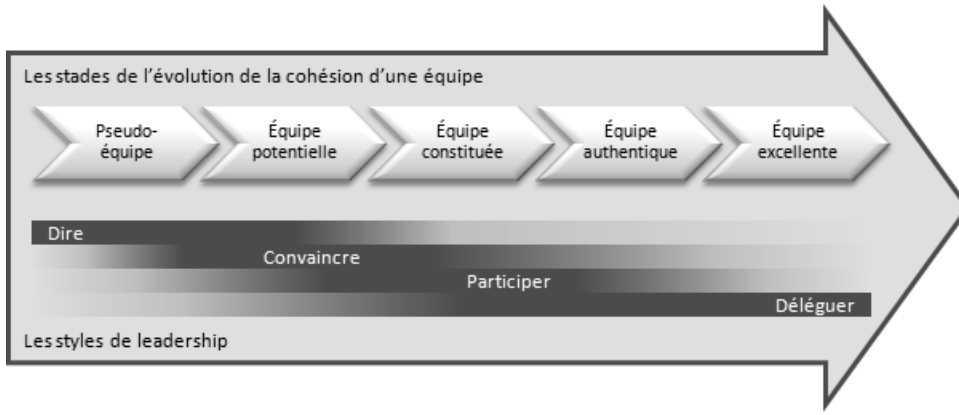


Figure 5.5 — L'évolution de la cohésion d'une équipe, selon Devillard, couplée aux styles de leadership.

À chacun des stades de l'évolution de la cohésion correspond un certain nombre de caractéristiques[2] :

- **Pseudo-équipe** :
  - Individualisme et compartimentation ;
  - Sentiment d'appartenance et de coresponsabilité faible ;
  - Relation un à un avec le responsable.
- **Équipe potentielle** :
  - Cohésion au plan interpersonnel seulement ;
  - Sentiment d'appartenance.
- **Équipe constituée** :
  - Les objectifs individuels s'inscrivent dans ceux de l'équipe ;
  - Cohésion interpersonnelle et opérationnelle.
- **Équipe authentique** :
  - Chacun conduit ses actions en relation continue avec celles des autres ;
  - Souci permanent de cohésion et cohérence.
- **Équipe excellente** :

1. Inspiré d'un billet de Martin Proulx, *I don't believe in self-organized teams…*, http://analytical-mind.com/2010/08/30/i-don%E2%80%99t-believe-in-self-organized-teams%E2%80%A6/, consulté en mars 2011.
2. Tiré d'un billet de Gaillard conseil, *Stades de maturité d'équipes*, http://www.gaillard-conseil.com/, consulté en mars 2011.

– L'intelligence collective de l'équipe lui permet une réaction unitaire dans toutes les situations.

5.4.1 Pourquoi?

Dans un processus humain, la motivation des individus reste encore le plus grand facteur de la productivité. Plusieurs ont cherché les meilleurs moyens pour motiver les humains et ont conclu que les facteurs comme la rémunération, les récompenses et la sécurité d'emploi, bien qu'ils soient importants, ont un impact limité sur la motivation. Les facteurs motivants, au point d'encourager le dépassement de soi, demeurent le sentiment d'accomplissement et l'engagement des individus. C'est pourquoi les organisations ont avantage à investir dans la création d'équipes motivées et autogérées.

> **Principe agile :** Bâtissez le projet autour de personnes motivées. Donnez-leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail.

Ce principe traduit bien ce que les approches agiles entendent lorsqu'il est question de former des équipes motivées et autogérées.

5.4.2 Comment ?

Toutes les équipes et toutes les organisations ne sont pas au même niveau du modèle proposé par Devillard. Bien que certaines parviennent à former des équipes excellentes rapidement, il est possible que d'autres n'y parviennent qu'après plusieurs années d'investissement et de changement de culture.

La présente sectionne traite pas des façons d'accompagner des équipes à devenir excellente. Mais nous proposons quelques considérations[1] pour aider à créer des milieux de travail où la formation d'équipes motivées et autogérées est possible.

Instaurer un climat de confiance

> « *L'opposé de l'autonomie est le contrôle. Le contrôle mène à la conformité ; l'autonomie mène à l'engagement.* »
> Daniel PINK[2]

Les valeurs de transparence ne sont possibles que si un climat de confiance existe, entre les membres de l'équipe et entre les équipes et les gestionnaires. Cela demande à chaque individu d'accorder le bénéfice du doute lorsqu'il analyse les décisions de ses collègues. Le principe est que ce sont des professionnels et que le résultat obtenu est ce qu'ils ont pu faire de mieux avec ce qu'ils avaient. Cela demande également de donner

1. Inspiré de l'ouvrage de Rachel Davis et Liz Sedley, *Agile Coaching*, Pragmatic Bookshelf, 2009.
2. Dans son ouvrage *Drive: The Surprising Truth About What Motivates Us*, Penguin Group, 2009.

le droit à l'erreur et de ne pas faire planer une épée de Damoclès au-dessus de la tête des individus, en châtiant chaque attente déçue et en rabâchant les erreurs du passé.

C'est seulement après avoir instauré ce climat de confiance que les individus et les équipes auront le courage de faire des demandes d'aide, de s'engager et de faire preuve de transparence.

Réunir les bons individus et construire des équipes multidisciplinaires

L'engagement et l'autonomie sont possibles lorsque les membres de l'équipe ne dépendent pas de personnes extérieures à eux. Cela amène à considérer certains aspects au moment de constituer l'équipe, comme de réunir toutes les compétences requises à la réalisation du projet et d'éviter la constitution d'une équipe tellement grande qu'il devient impossible de collaborer efficacement. Ensuite, le fait d'impliquer les clients ou les responsables d'affaires dès le début favorise la prise d'engagements par l'équipe. Pendant le projet, l'organisation qui valorise les équipes multidisciplinaires soutient les pratiques permettant le transfert de connaissances, comme le binômage et la conception en groupe.

Finalement, l'organisation doit éviter de considérer les individus comme des ressources pouvant être déplacées d'une équipe à l'autre. Il faut plutôt favoriser la protection des équipes pour les encourager à partager leurs compétences et devenir prévisibles et efficaces ensemble. Un billet de Martin Proulx[1] encourage même à constituer des équipes qui, au lieu d'être démantelées au terme d'un projet, seraient plutôt affectées à un nouveau projet (fondé sur la pratique qu'une équipe gagnante, comme les champions du *Super Bowl* ou du *Mundial*, n'est pas démantelée au terme de la saison).

C'est un des changements importants. Les organisations d'aujourd'hui ne sont pas contre la notion d'équipe. Toutefois, les processus d'affectation à des projets doivent être significativement changés pour prendre en compte la notion de cohésion d'équipe. Aujourd'hui, dans les grandes organisations, nous voyons pratiquement que des pseudos équipes, de temps en temps des équipes potentielles et que très rarement des équipes constituées, authentiques ou excellentes. Le démantèlement des équipes provoque une remise à niveau de leur maturité et leur cheminement est à recommencer. L'organisation ne profite alors jamais de l'investissement qu'elle fait envers l'objectif de former des équipes performantes.

Encourager l'engagement

Les approches agiles préconisent un rythme de travail soutenu et soutenable. Cela ne veut pas dire qu'il est acceptable qu'une équipe sous-évalue sa capacité. Les individus sont motivés lorsque la charge n'est pas démesurée et exténuante et lorsqu'elle n'est pas insignifiante et ennuyante. Ensemble, l'organisation et les équipes doivent trouver

1. Billet de Martin Proulx, *Great news the project is over! Now let's dismantle the team*, http://analytical-mind.com/2011/02/15/great-news-the-project-is-over-now-lets-dismantle-the-team/, consulté en février 2011.

un juste milieu. Un autre moyen d'encourager l'engagement est de clairement indiquer aux équipes l'importance de leurs projets et la place qu'ils tiennent dans la stratégie de l'entreprise.

Finalement, il est bénéfique de donner de la reconnaissance aux équipes, de donner de la visibilité aux succès des projets et de célébrer les accomplissements, afin d'exprimer la fierté de travailler ensemble. Il faut cependant se méfier des récompenses amenant la compétition entre les individus et entravant l'esprit d'équipe, comme les bonus strictement individuels.

Créer un environnement de travail motivant et innovant

Plusieurs facteurs différents peuvent influencer la perception qu'ont les individus envers leur espace de travail :

- **La qualité de vie** : un réseau informatique défaillant, des postes de travail non-performants et l'indisponibilité de la machine à café ne sont pas des facteurs de motivation en soit, mais font partie des conditions implicites qu'un professionnel peut s'attendre de son environnement de travail.
- **Un espace de travail réunissant les membres d'une équipe dans un même espace**, sans murs ni obstacles entre les personnes. Un tel espace permettant d'afficher les indicateurs de projet sur les murs et de concevoir ensemble sur un tableau blanc est un exemple d'un environnement favorisant la création de d'équipes collaboratives. Si des contraintes physiques l'empêchent, essayez graduellement de rapprocher les membres d'une même équipe sur le même étage car la proximité est un facteur favorisant le développement de la cohésion d'une équipe. Si vous devez mettre en place des équipes distribuées géographiquement, nous recommandons de les réunir au même endroit en début de projet, le temps nécessaire pour qu'une cohésion d'équipe prenne forme.
- **Laisser place à l'innovation** : en réservant des moments où les individus peuvent faire des preuves de concepts et réaliser des projets personnels, l'organisation encourage le dépassement de soi et la recherche de l'excellence.
- **Favoriser l'amélioration des compétences** : des mécanismes peuvent être mis en place, tel que le compagnonnage et la création de communautés de pratiques.

Fixer la taille de l'équipe

La taille d'une équipe autogérée est un facteur important de son efficacité et elle doit tenir compte des aspects suivants :

- **Permettre à la collectivité de travailler de manière efficace :** il est plus difficile de se faire entendre et faire consensus dans un grand groupe qu'un petit.
- **Favoriser l'engagement :** dans une grande équipe, un membre peut limiter sa contribution car il a un faux sentiment de confiance que quelqu'un d'autre exécutera le travail. Cela est d'autant plus observable lorsque la personne est spécialisée, se contentant d'exécuter les tâches liées à sa spécialité.

- **Resserrer les liens entre les membres :** il est plus facile d'entretenir des liens de confiance et de la complicité avec tous les individus d'une petite équipe. Les grands groupes ont plus de chance de s'organiser en sous-groupes.

Pour toutes ces raisons, les méthodes agiles proposent que la taille des équipes se situe entre cinq et neuf individus (7±2). Ce n'est évidemment pas une règle absolue, mais l'expérience prouve que cette taille est particulièrement efficace pour favoriser la collaboration et l'engagement de tous ses membres. Au-delà de cette taille, il est conseillé de diviser l'équipe en plusieurs groupes différents.

Éviter les ressources partagées

La gestion du temps personnel est un défi pour n'importe quel professionnel et les approches agiles n'y échappent pas. Bien que nous soyons en mesure de faire simultanément plusieurs activités, c'est contre-productif. C'est une mauvaise habitude connue et largement documentée. La principale raison est le temps nécessaire afin de délester la charge cognitive de la première activité avant de charger les informations concernant la tâche suivante. Il est donc plus efficace de se concentrer sur un seul engagement plutôt que de « se remettre dans le bain » perpétuellement.

De plus, les engagements multiples nous rendent imprévisibles. Même s'il est possible de planifier la proportion de temps disponible pour chacune des activités, il y a de fortes chances pour que les imprévus d'une activité consomment le temps originalement prévu pour les autres. Il est utopique de penser qu'aucun imprévu ne puisse changer les priorités de quelqu'un responsable de plusieurs activités en simultané.

Les engagements d'une équipe sont basés sur leur performance collective et lorsque la disponibilité de ses membres est chaotique, l'équipe ne parvient plus à les tenir. Conjointement, ils peuvent réaliser en parallèle les activités d'une même itération. Lorsque chaque membre participe à plusieurs projets, cela limite le travail en collaboration. L'équipe devenant alors un collectif d'individus n'étant plus imputable en groupe.

5.4.3 Contributions et impacts sur les différents corps de métiers

Les développeurs

Les développeurs sont responsables d'estimer l'effort pour réaliser chacune des spécifications, de planifier l'exécution des itérations, et de démontrer le résultat de leurs travaux. Toutes ces responsabilités sont collectives. Ensemble, les membres sont en mesure de s'engager, d'estimer, de planifier, de réaliser et de démontrer. Ils profitent de leur intelligence collective et ne craignent pas de s'afficher parce qu'ils sont tenus responsables collectivement de leurs actions et de leurs résultats.

Plusieurs de ces responsabilités sont traditionnellement attribuées au rôle de chargé de projet. Si cette transformation est récente dans votre organisation, il est très probable que vous soyez préoccupé à l'idée que l'équipe de projet ne soit pas en mesure de prendre ces responsabilités. Faire accompagner l'équipe par un coach d'équipe

est une bonne façon de réduire ces préoccupations. Ce coach doit avoir les mêmes préoccupations que peut avoir un chargé de projet et être capable d'accompagner l'équipe pour qu'elle devienne autonome et performante.

Le client

Le rôle du client dépasse largement celui du simple expert d'affaires. Ce rôle s'approche plutôt des responsabilités d'un directeur produit et il est attendu que le client soit en mesure de :

- communiquer la vision du produit ;
- réunir les besoins d'affaires ;
- entretenir le carnet de produit ;
- planifier les itérations ;
- évaluer les incréments de produit.

La plupart des clients que nous rencontrons n'ont pas de difficulté à expliquer la vision et recueillir les besoins d'affaires, parce qu'ils sont souvent experts du domaine d'affaires. Votre organisation dispose-t-elle de personnes ayant un profil de directeur produit en mesure de spécifier, valider et planifier un projet de développement ? Ces activités sont exigeantes et les clients sont souvent surpris de découvrir l'ampleur de la charge de travail qui leur incombe.

Cette charge de travail peut être amoindrie en affectant un client à temps plein avec l'équipe de projet et en l'entourant de spécialistes tels que des analyses ou un coach agile pour compléter leur profil de compétences.

Les analystes

Qu'ils proviennent du domaine d'affaires ou du département technique, les analystes contribuent à améliorer la compréhension et la collaboration entre le client et l'équipe de projet.

Les analystes ont le sens de l'écoute des clients et ont de l'intérêt pour le travail des utilisateurs finaux. Ils ont des préoccupations d'utilisabilité, de cohérence, de simplicité, d'efficacité envers les logiciels développés.

Dans une approche traditionnelle, les analystes étaient souvent responsables de réaliser l'analyse des besoins avec les clients. Cette analyse était ensuite traduite en spécifications détaillées pour le développement de la solution. Avec une approche agile, l'analyste travaille avec le reste de l'équipe :

- Il aide le client avec l'entretien du carnet de produit, en complétant les spécifications avec les éléments plus techniques tels que les diagrammes des processus, les règles d'affaires, les prototypes d'écrans.
- Il conseille le client avec la priorisation du carnet de produit pour assurer le développement d'un système cohérent.

- Il veille à limiter l'analyse du système afin de garder une avance suffisante permettant d'alimenter les équipes de projet, en évitant le piège de l'analyse détaillée préliminaire.

Pour éviter de devenir un goulot d'étranglement, l'analyste doit apprendre à partager son expertise, à collaborer avec le reste de l'équipe et à profiter de l'intelligence collective du groupe.

Les testeurs et les responsables de l'assurance qualité

Dans une approche agile, l'assurance qualité est l'affaire de toute l'équipe. Comme pour les analystes, les responsables de l'assurance qualité (QA) doivent éviter de devenir les goulets d'étranglement de l'équipe. Dans leur ouvrage *Agile Testing*, Lisa Crispin et Janet Gregory donnent des conseils sur l'attitude à adopter :

> *« Ne restez pas dans l'attente, soyez proactifs ! Nous les testeurs ne pouvons pas attendre que les tâches de tests nous parviennent. Nous devons être impliqués et participer à l'identification des travaux de tests. La collaboration avec les programmeurs est un nouveau concept pour plusieurs testeurs. La collaboration avec les clients aussi. Cela oblige de sortir de sa zone de confort. Les programmeurs sont des personnes occupées et ils peuvent être intimidants. Lorsque j'ai débuté en tant que testeur d'une équipe de huit programmeurs, même si j'avais antérieurement travaillé avec la plupart de mes coéquipiers dans d'autres projets, j'avais tout de même besoin de courage pour leur demander leur aide.*[1] »

Le défi pour les responsables QA est de passer d'un mode de travail séquentiel à un mode collaboratif partagé et itératif. La précédente section *La stratégie de tests* propose des pistes de réflexions en ce sens.

Les gestionnaires

Après avoir compris le concept d'équipe autogérée, le gestionnaire qui le met en œuvre transfère une partie de son autorité et de ses responsabilités aux équipes afin qu'elles puissent effectuer leur travail de manière plus autonome. Le gestionnaire doit alors éviter l'ingérence dans les équipes et la micro-gestion de leurs activités car il donne alors l'impression que l'autogestion n'est pas réelle. S'il décide d'observer les pratiques agiles des équipes, il devrait le faire de manière transparente et en discuter avec le coach d'équipe afin de s'assurer que sa seule présence ne gêne pas les intervenants et qu'il comprend les décisions prises par l'équipe.

Ce comportement devrait être possible dans les organisations où le droit à l'erreur existe, c'est-à-dire celles dans lesquelles les erreurs sont vues comme des expériences permettant à tous les intervenants de tirer des leçons. Plutôt que de blâmer des individus, le droit à l'erreur donne le courage aux individus d'être transparents, une condition nécessaire à l'amélioration collective des processus. Cette attitude

1. Traduction française libre, tirée du livre de Lisa Crispin et Janet Gregory, *Agile Testing: A Practical Guide for Testers and Agile Teams*, Addison-Wesley, 2009.

soutient les mécanismes d'apprentissage et d'amélioration et évite de pénaliser systématiquement les défaillances.

Toutefois, il peut être attendu qu'une équipe nouvellement formée n'ait pas au départ la maturité nécessaire à l'autonomie souhaitée. C'est dans ces cas qu'un style de leadership adaptatif doit être appliqué par le gestionnaire qui doit veiller à se comporter pour favoriser la montée en maturité de l'équipe.

En résumé

Nous avons fait la distinction entre la planification incrémentale et le suivi d'un plan de livraison strict. La planification incrémentale est bien adaptée au développement logiciel car elle prévoit l'ajustement des leviers de gestion de projet à chaque itération. Cependant elle demande de la rigueur pour ne pas augmenter le risque de s'enthousiasmer autour de la construction d'un produit prometteur mais inutile ou incohérent au final.

Les spécifications composant le carnet de produit décrivent plus souvent des fonctionnalités que des travaux techniques. C'est pourquoi le client doit participer à la planification des travaux techniques en les incluant à l'intérieur des demandes fonctionnelles. La fusion entre les besoins techniques et d'affaires permet d'avoir une meilleure mesure de l'état d'avancement du projet au fil des itérations.

Pour pouvoir s'engager à la livraison d'incréments de logiciel complet, une équipe doit être multidisciplinaire. La constitution d'une telle équipe a des impacts sur la collaboration interne des membres et sur les interventions des spécialistes ne pouvant s'intégrer à l'équipe à temps complet.

Nous avons expliqué comment les itérations peuvent être utilisées comme des points de visibilité. Cependant, la fréquence de ces points n'est pas le seul élément à considérer lors de l'établissement de la longueur des itérations. Il faut aussi considérer la capacité de l'équipe à livrer des incréments de logiciel de qualité « production ». Les itérations ne sont pas seulement des points de visibilité. Elles donnent l'occasion de tenir régulièrement des activités d'introspection afin d'améliorer les façons de faire de l'équipe et de son organisation.

6

Gestion de projet

Objectif

Suite à la lecture de ce chapitre, vous constaterez combien, avec les approches agiles, les affaires tiennent un rôle important dans la gestion des projets de développement. Vous serez en mesure de distinguer les approches de gestion de projet déterministes de celles empiriques. Vous serez aussi en mesure de mesurer la progression des projets par des indicateurs différents de ceux du budget et de l'échéancier et de décrire les rôles des intervenants concernés par la gestion de projet. Nous proposons également quelques gabarits de diagrammes d'état d'avancement, construits à partir d'indicateurs adaptés au contexte particulier des projets.

Mise en situation : Jacinthe, Cynthia et l'indicateur de progression

Une compagnie d'assurance désirant réduire le temps de traitement des demandes de réclamation aux particuliers a demandé une étude à un expert en processus du travail afin d'identifier les optimisations possibles. Un grand nombre de propositions de l'expert concernent l'amélioration des différents systèmes informatiques utilisés par les conseillers aux réclamations. La compagnie, désirant obtenir les gains proposés par le rapport d'expertise, a décidé d'investir dans un projet d'évolution applicative totalisant 2 000 jours-personnes.

Jacinthe et Cynthia, des conseillères seniors de l'équipe des réclamations aux particuliers, sont conjointement responsables de la planification du projet. À partir du rapport d'expertise, elles ont rédigé un carnet de produit avec, comme critères de priorisation, le gain potentiel de réduction des temps de traitement. Leur objectif est de planifier en priorité les gains les plus importants de façon à maximiser le retour sur investissement pour chacune des livraisons.

Daniel, leur directeur, désire suivre la progression du projet. Il leur demande de remettre, au terme de chaque itération, un état d'avancement du projet. À titre d'exemple, il leur remet une version du graphique utilisée dans un précédent projet, décrivant le budget consommé en fonction de l'échéancier. Cynthia demande s'il est possible d'utiliser un modèle différent.

Cynthia : Ce graphique mesure la consommation des ressources, mais ne démontre pas les gains obtenus. Daniel, trouverais-tu utile de recevoir un graphique qui illustre les gains obtenus d'une itération à l'autre ?

Daniel : Bien sûr, il serait pertinent de démontrer au reste du comité de direction la valeur ajoutée du développement. Cependant, le comité désirera tout de même mesurer la consommation des budgets selon l'avancement du projet.

Cynthia : Je comprends. Par contre, la consommation du budget sera linéaire, puisque le nombre de personnes affectées restera constant pendant le projet. Le bénéfice que Jacinthe et moi visons, avec notre planification, sera plus logarithmique que linéaire. Je propose de concevoir un modèle différent qui pourra à la fois mesurer les gains obtenus et la consommation des ressources.

Daniel : Pensez alors à me présenter votre nouveau modèle avant la prochaine rencontre du comité. Cela me permettra de préparer une introduction qui facilitera l'adoption du graphique par les autres directeurs.

6.1 LES APPROCHES DE GESTION

Un impact important de l'agilité sur la gestion de projet, c'est le choix entre l'approche déterministe et l'approche empirique. Nous utiliserons le jeu du golf comme analogie pour faire la distinction entre les deux approches, et ensuite expliquer comment l'approche empirique se marie mieux à un contexte de gestion agile que celle déterministe.

Les objectifs du golf sont très simples : avec l'utilisation d'un club, le joueur doit mettre une balle dans un trou situé tout au bout du terrain et ce, pour chacun des trous du parcours. Même si les objectifs sont simples, l'exécution l'est moins. Au départ du parcours, le joueur doit choisir le club en fonction des facteurs suivants : la distance à parcourir, l'habilité du joueur à frapper la balle, la forme du terrain et les conditions météorologiques. En fonction du résultat obtenu sur ce premier coup, le joueur doit choisir à nouveau un club pour son prochain coup en réévaluant tous les facteurs. Après chacun des coups, le joueur devrait être de plus en plus près de son objectif. Aussi, plus la partie progresse, plus la compréhension du terrain s'affine pour le joueur et plus l'évaluation de ses capacités est juste.

6.1.1 L'approche déterministe

Essayons de planifier une partie de golf de 18 trous avec le moins d'imprévus possible. En commençant par le premier parcours d'une normale de trois coups, nous pourrions prévoir qu'il est possible de réussir avec un « Bois #3 », un « Fer #7 » et le « *Putter* ». Et nous pourrions continuer de la même manière pour les 17 autres trous. Au final,

nous aurions la liste complète de toutes les activités de la partie et une estimation de notre score final. Nous aurions ainsi un plan précis à exécuter basé sur un modèle déterministe.

Cependant, n'importe quel imprévu pourrait mettre le plan en péril. Par exemple, comment faire pour savoir avec certitude l'endroit où se situera la balle au coup suivant ? Est-ce que notre plan tient compte de l'expérience du joueur ? Les conditions météorologiques de la journée sont-elles prises en compte ? Il suffit que l'une des activités prévues dans notre plan ne se déroule pas comme prévu pour que le plan doive être modifié. Plus un plan est précis, moins il tolère les erreurs.

Cette façon de prévoir une partie de golf ressemble aux projets de développement souvent illustrés sous forme de diagrammes de Gantt. Dans un diagramme de Gantt, toutes les activités sont bien détaillées, estimées et séquencées de manière à pouvoir prédire précisément tout ce qui doit se produire, à quel moment et par quels intervenants. C'est une façon de s'assurer que toutes les activités nécessaires pour atteindre les objectifs du projet tiendront à l'intérieur des contraintes de temps et de budget. Poussée à l'extrême, cette stratégie laisse peu de place aux imprévus qui pourraient survenir et compromettre le bon déroulement du plan. Pour réussir avec un plan de type déterministe, il faut prévoir une marge de manœuvre qui absorbera les imprévus possibles. Et c'est là qu'un responsable chargé de la gestion de projet doit maîtriser l'art de gérer les risques, en incluant dans son plan l'ensemble des activités pour atténuer et contingenter les risques connus et les imprévus qui n'ont pas encore été identifiés. Mais si certains arrivent à identifier la majorité des risques en début de projet, peu de gestionnaires peuvent prétendent avoir estimé convenablement l'effort requis pour les activités d'atténuation et de contingence. Cette sous-estimation est en partie la cause des dépassements budgétaires affectant la majorité des projets de développement logiciel[1].

6.1.2 L'approche empirique

Il est difficile, voire impossible, d'estimer précisément le nombre de coups nécessaires pour terminer une partie de golf, car de nombreux facteurs influencent les résultats. Le golf n'est pas prédictif : il est plutôt basé sur les observations et l'expérience du joueur, pendant toute la partie. Un joueur de golf avec une stratégie empirique obtiendra donc de meilleurs résultats qu'avec une stratégie déterministe. Le dictionnaire Larousse définit le mot empirique comme suit :

> **Empirique** : Qui ne s'appuie que sur l'expérience, l'observation.

Pour la gestion des projets de développement logiciel, le choix entre une approche empirique ou déterministe dépend du niveau de confiance face aux facteurs d'incertitude. En général, les facteurs d'incertitude d'un projet de développement logiciel sont les technologies utilisées, la maîtrise du domaine d'affaires et les ressources humaines affectées. Dans son livre *Strategic Management and Organisational Dynamics :*

1. Source : le *Chaos report* par le *Standish Group*, de 2004 à 2009.

The Challenge of Complexity, Ralph D. Stacey représente les facteurs des technologies et du domaine dans un graphique illustrant la complexité des projets de développement (Figure 6.1).

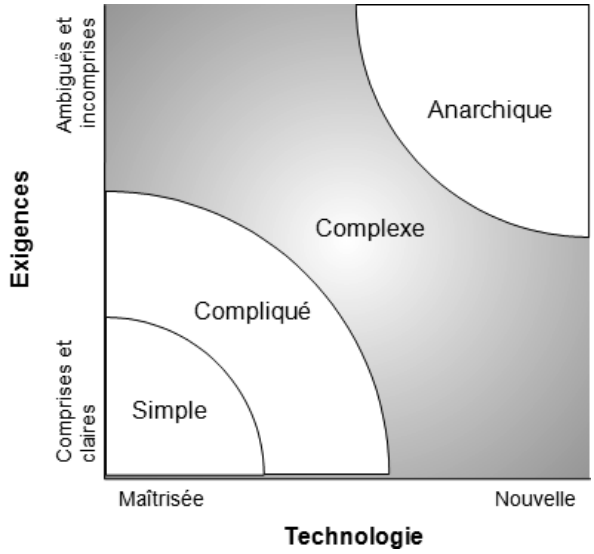


Figure 6.1 — Complexité des projets en fonction de la maîtrise du domaine d'affaires et des technologies.

Une approche de gestion déterministe est adéquate pour des projets simples, c'est-à-dire dont les technologies utilisées sont maîtrisées et dont le domaine d'affaires associé est bien connu. Par contre, s'il existe de l'incertitude autour d'un de ces deux facteurs, il est préférable d'utiliser une approche empirique. Trop souvent, les projets de développement logiciel sont évalués de façon optimiste. En outre parce que le facteur humain s'ajoute aux deux précédents facteurs et il est difficile au départ de quantifier son influence sur l'estimation.

Même lorsque les membres d'une équipe parviennent à réunir toutes les compétences technologiques et les connaissances associées au domaine, rien ne garantit qu'ils soient efficaces pour travailler ensemble. Le développement logiciel n'est pas une science exacte : c'est un processus créatif dont le succès relève grandement des individus impliqués.

En résumé, les facteurs technologiques, humains et du domaine d'affaires complexifient la planification des projets de développement. Ce qui fait qu'une gestion par approche empirique est généralement mieux adaptée aux projets de développement logiciel.

6.1.3 Passer d'une approche déterministe à une approche empirique

> **Principe agile :** Les processus agiles exploitent le changement comme avantage compétitif pour le client.

Plusieurs organisations précèdent le lancement de leur projet par une étude d'opportunité permettant d'en évaluer la faisabilité et la rentabilité. Le calcul de la rentabilité commande l'estimation des ressources nécessaires à la réalisation du projet. Les coûts et les délais de projet sont calculés à partir des efforts estimés pour livrer les fonctionnalités et les caractéristiques. Le résultat de l'étude d'opportunité sert ensuite d'intrant à la décision de lancer ou non le projet.

Une fois le lancement autorisé suite à une étude d'opportunité concluante, les délais et les coûts estimés sont connus pour le développement de la portée du projet. Une approche déterministe fait l'hypothèse que ces estimations sont réalistes et qu'il est possible de faire le suivi du projet en mesurant la consommation des ressources. C'est possible tant que les ressources sont suffisantes pour couvrir le développement de l'ensemble des exigences et qu'elles ne changent pas pendant le déroulement du projet.

Mais pour permettre le changement, développer de nouvelles fonctionnalités, s'ajuster aux imprévus et remettre en question certaines exigences en cours de projet, il faut adopter une approche empirique. Et dans ce cas, l'hypothèse des ressources réalistes ne tient plus, car chaque changement, ajout ou suppression aux exigences entraînera forcément une modification aux ressources requises. Du coup, les ressources consommées ne peuvent plus servir de seule base de calcul pour le suivi de projet. Ce suivi doit être fait en mesurant les variations des exigences. Dans ce cas, la stratégie de planification est d'optimiser le développement d'exigences ayant la plus grande valeur d'affaires dans les limites des ressources disponibles. Ce retournement de stratégie de planification et de suivi est illustré à la sous la forme de triangles inversés.

La gestion de projet basée sur le suivi d'un plan strict est représentée par le triangle de gauche de la Figure 6.2. Le suivi d'un plan strict s'accompagne habituellement de la mise en œuvre d'un processus pour gérer les changements aux exigences. Malheureusement, ce processus peut s'avérer assez lourd, car chaque modification implique de changer le plan et, dans bien des cas, le budget et l'échéancier.

La gestion de projet empirique se fait à partir d'une vision, de façon à livrer de la valeur d'affaires le plus tôt possible (triangle de droite de la Figure 6.2). Ici, ce n'est plus le suivi d'un plan rigide qui compte mais bien le développement empirique, où les exigences sont inspectées et modifiées périodiquement afin de combler les besoins d'affaires mieux connus ou changeant pendant le projet. Le changement est donc bienvenu, voire encouragé avec cette approche, dans la mesure des ressources disponibles.

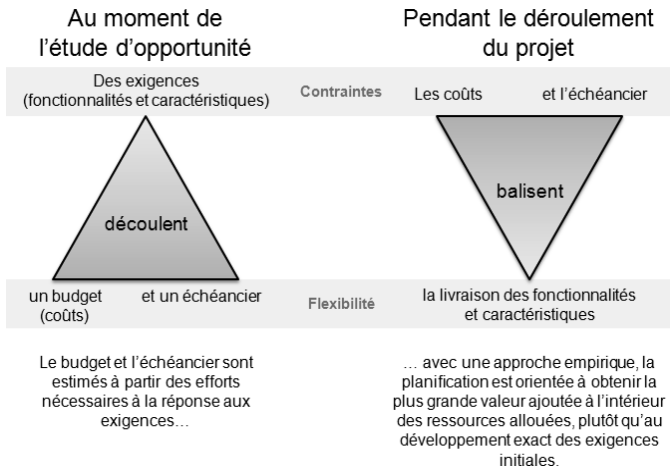


Figure 6.2 — Le budget, l'échéancier et le périmètre d'un projet sont abordés différemment avant et pendant un projet.

6.2 MESURER L'AVANCEMENT

> Phrase entendue dans une conversation sur la gestion des exigences : « *Marcher sur l'eau est tout aussi faisable que de livrer l'ensemble des exigences requises, dès l'instant où les deux sont gelés* ».

Les méthodes agiles précisent peu de mesures d'avancement, sinon que le logiciel fonctionnel est le meilleur moyen de mesurer l'avancement d'un projet. Cependant, elles ne limitent aucunement ce qui peut être mesuré. Toutes les mesures de l'avancement d'un projet doivent servir à prendre des décisions et à poser des actions lorsque le projet dévie du plan. Peu importe la nature du projet, il est fréquent d'en mesurer les facteurs suivants :

- les coûts en relation avec le budget ;
- les dates de livraison en relation avec l'échéancier ;
- la qualité en relation avec le niveau attendu ;
- les fonctionnalités et caractéristiques en relation avec la valeur d'affaires prévue et l'atteinte des objectifs d'affaires.

Ces quatre facteurs, illustrés aux cimes du tétraèdre (Figure 6.3), constituent les leviers sur lesquels le responsable de produit peut agir en cas de déviation et ils pourraient théoriquement tous être modifiés en cours de projet. Par ailleurs, ces quatre leviers peuvent aussi être ou devenir des contraintes lorsqu'ils sont fixés en tant qu'enjeux ou objectifs.

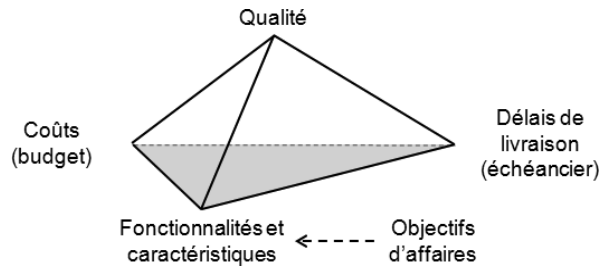


Figure 6.3 — Mesures et indicateurs classiques de la gestion de projet.

6.2.1 Les coûts et les délais

La base des mesures liées aux coûts et aux délais est différente selon que le projet est géré avec une approche déterministe ou empirique (Tableau 6.1).

Tableau 6.1 — Comparatif de la base des mesures de coûts et de délais d'un projet entre l'approche est déterministe et empirique.

Élément de comparaison	Approche déterministe	Approche empirique
Coûts mesurés par	Activité et livrable	Itération
Total des coûts par	Phase et projet	Livraison et projet
Mesures de tendance de la progression des coûts	Comparaison entre les coûts réels et le budget prévu des activités et livrables à une date donnée. Puis extrapolation sur les activités et livrables restants. L'hypothèse est que la liste des activités et livrables comble les besoins et qu'elle est planifiée de manière adéquate.	Comparaison entre les coûts réels et prévus à la fin d'une itération donnée. Puis extrapolation sur la taille des fonctionnalités terminées comparées à la taille des fonctionnalités restantes. Voir le Sunset Graph plus loin dans ce chapitre.
Délais mesurés par	Activité et livrable	La fin d'une itération
Jalons suivis par	Phase et projet	Itération, livraison et projet
Mesure de tendance de progression des délais	Comparaison à une date donnée entre les dates de livraison réelles et prévues des activités et livrables. L'hypothèse est que les délais établis sont adéquats et que les activités et livrables sont complets.	Extrapolation du nombre de la taille des fonctionnalités livrées à date sur la taille de celles restantes et comparaison avec la date ciblée. Voir le Sunset Graph.

6.2.2 La qualité : levier ou contrainte ?

Il n'est pas recommandé d'utiliser la qualité en tant que levier en cours de projet, même en ultime recours. Il est préférable d'en définir les critères réalistes au moment du démarrage et d'en faire le suivi pendant le déroulement du projet.

La mesure de la qualité en relation avec le niveau attendu consiste à passer en revue chacun des critères de la définition de terminé avec les activités et livrables

accomplis pendant une ou plusieurs itérations. La définition de terminé sert alors de liste de vérification où chacun des items sera complété ou non, sans ambiguïté.

Autrement, pour chacun des items non complétés de la liste, une dette technique[1] s'accumule. Sa valeur équivaut à l'effort requis pour compléter l'ensemble des items incomplets auquel s'ajoute l'effort de contingence et d'atténuation des risques liés à la non-qualité. Il est souvent plus sage d'avoir une définition de terminé tenant compte de ces risques. Il est généralement moins coûteux d'assurer la qualité tout au long du projet conformément à cette définition de terminé que d'avoir à gérer les risques de ne pas s'y être conformé.

6.2.3 Les fonctionnalités et caractéristiques

La contrainte sur la qualité est essentielle en cours de projet pour mesurer objectivement l'avancement sur les fonctionnalités et caractéristiques. Autrement, cette mesure d'avancement ne serait pas uniforme, car la dette technique potentiellement accumulée par endroits créerait une différence.

La progression du développement des fonctionnalités et caractéristiques se mesure en comparant la taille ou la valeur des items complétés du carnet de produit avec la taille ou la valeur des items restant. Plusieurs mécanismes simples de suivi sont proposés à la section *Les livrables* plus loin dans ce chapitre.

6.2.4 L'atteinte des objectifs d'affaires

Peu importe l'approche de gestion appliquée, l'atteinte des objectifs d'affaires d'une solution logicielle commence à se mesurer avec le premier déploiement. Dans une approche déterministe, le déploiement est souvent unique, à la fin du projet. Il est alors impossible de suivre directement ce facteur pendant le déroulement du projet.

Dans une approche empirique et agile, le projet est découpé en plusieurs livraisons. En cours de projet, il est alors possible de déployer en production et débuter la mesure de l'atteinte des objectifs d'affaires dès que l'application est utilisée, tel que traité à la section *La capitalisation*.

Le résultat d'un projet devrait donc se mesurer à la qualité de la solution logicielle déployée tandis que le budget et l'échéancier doivent être considérés comme des contraintes à respecter et à suivre plutôt que des mesures de progression.

1. La dette technique est une analogie illustrant le travail s'accumulant de façon insidieuse pendant le développement d'un produit. Pour plus d'informations à propos de la dette technique, nous vous invitons à consulter le chapitre *5 Équipes et livraison incrémentale*.

6.2.5 La valeur acquise

La problématique de la méthode de la valeur acquise traditionnelle

En gestion de projet, le concept de la « valeur acquise »[1] représente la portion du budget et de l'échéancier consommé à partir des tâches et livrables prévues et réalisées dans le plan de projet à un moment donné. Dans un projet non agile, il n'est pas rare que plus de la première moitié de ces tâches et livrables constituent des documents de planification, de conception et d'analyse. Quand ces tâches et livrables sont complétés, les portions consommées du budget et de l'échéancier sont utilisées pour calculer la valeur acquise. Mais à ce stade du projet, qu'y a-t-il d'acquis ? Des documents ont été livrés, des ateliers de travail et des réunions ont été tenus, des tâches ont été accomplies, le budget a été dépensé pendant que le temps s'écoulait et ce, sans nécessairement livrer de réelle valeur pour le client. Le projet a accumulé un inventaire qui serait possiblement sans valeur si le projet était arrêté. En conséquence, plutôt que de servir à mesurer l'état d'avancement, la valeur acquise réduit la visibilité sur le déroulement du projet. Pour vérifier que le projet s'est bien déroulé, il faut évaluer le résultat des premières fonctionnalités livrées. Avec un cycle de vie en cascades, cela n'est possible que tard dans le déroulement du projet, parfois à 80 % du terme prévu. Dans le cas où le résultat serait insatisfaisant, le plan devra être modifié pour redresser la situation. Malheureusement, plus le résultat de cette vérification tarde, plus les impacts sur le budget et l'échéancier risquent d'être grand. Étant donné une plus grande quantité de logiciel développé, la quantité de travaux antérieurs devant être repris est également plus grande.

> **Principe agile** : Un logiciel fonctionnel est la meilleure unité de mesure de la progression du projet.

En mode agile, parce que les activités sont réparties à travers toute la durée du projet, la valeur acquise calculée à partir des activités et livrables est inadéquate et non souhaitable Par exemple, il peut être possible qu'un dossier fonctionnel, commencé au début du projet, soit complété seulement à la livraison finale, lorsque toutes les options souhaitées par le client auront été développées. Dans ce cas, la valeur acquise déterminée par l'effort consacré au dossier fonctionnel pendant le projet n'est pas pertinente, en plus d'être fausse d'un point de vue étymologique. Il en va de même pour toutes les activités distribuées dans plusieurs itérations, où seul le suivi de l'effort restant est assuré par les membres de l'équipe.

Nous ne déclarons pas que le suivi des coûts et des délais est à éliminer, mais qu'il est simplement insuffisant. Malgré tout certains gestionnaires sont réfractaires à l'idée d'adopter de nouveaux indicateurs de suivi des projets :

- parce qu'ils n'en connaissent pas de meilleurs ;
- parce que l'organisation impose des indicateurs normalisés.

1. Traduit de l'anglais *Earned Value*. Pour plus d'informations sur la méthode de la valeur acquise (*Earned Value Management*), vous référer àhttp://fr.wikipedia.org/wiki/Gestion_de_la_valeur_acquise.

Transformer la façon de calculer la valeur acquise

Plusieurs façons de calculer la valeur acquise peuvent être envisagées lorsque l'organisation tient absolument à conserver cet indicateur. Tout dépend des besoins ayant trait à la présentation des résultats de suivi et aux décisions à prendre sur les leviers de réalignement. Ainsi, dans un projet suscitant de grandes attentes quant à la valeur d'affaires de ses fonctionnalités, la valeur acquise pourrait être calculée par la somme de la valeur d'affaires de tous les items livrés du carnet de produit à la fin d'une itération ou d'une livraison donnée. Cette valeur est alors transformée d'un ratio (façon traditionnelle pour illustrer la valeur acquise) à une valeur monétaire. Elle pourrait être présentée sous forme de ratio en la comparant avec la valeur d'affaires des fonctionnalités restantes.

Dans le contexte où le respect du budget est un enjeu, la valeur acquise peut être calculée à partir du ratio entre la taille des fonctionnalités livrées et celle des fonctionnalités restant à livrer, combiné au ratio du budget dépensé.

6.3 LES RÉPERCUSSIONS SUR LES INTERVENANTS

Nous avons vu comment les approches agiles et les approches empiriques transforment la façon de planifier et de suivre les projets de développement. Cette transformation introduit les concepts de l'inspection des livraisons d'incréments fonctionnels et de suivi avec des indicateurs mesurant la valeur d'affaires. Ce sont des concepts requérant une étroite collaboration avec les gens d'affaires qui sont les mieux qualifiés pour mesurer et suivre la valeur d'affaires.

En addition avec l'introduction d'une approche empirique, un autre impact important de l'adoption des approches agiles est l'introduction d'un responsable affaires au sein de la gestion des projets et la révision des rôles et responsabilité des chargés de projet de l'organisation.

6.3.1 Transfert de responsabilité vers le responsable de produit

> **Principe agile** : Les experts métier et les développeurs doivent collaborer quotidiennement au projet.

L'étroite collaboration des affaires au sein des équipes de projets peut provoquer un impact très fort sur le processus de gestion de projet. Cela est particulièrement vrai avec la méthode Scrum, où la responsabilité de gérer le projet de développement est grandement partagée avec le responsable de produit. Concrètement, le responsable de produit est en charge de :

- recueillir et analyser les besoins d'affaires ;
- écrire les exigences ;
- planifier la réalisation des exigences ;
- valider le résultat des incréments de logiciels.

Cela peut décontenancer les organisations qui confient normalement ces activités à un chargé de projet. En premier lieu parce que cela peut remettre en cause l'utilité des chargés de projets et favoriser l'introduction de coaches au sein des équipes. Mais surtout parce qu'un responsable affaires n'a pas la même perspective sur les facteurs de priorisation qu'un membre des TI. Un chargé de projet TI aura tendance à planifier un projet de développement en divisant le travail à accomplir en tâches (comme l'installation d'une base de données), en livrables techniques (comme la rédaction d'un dossier fonctionnel) et mesurera la réussite d'un projet à la date de livraison de la solution logicielle. Un responsable de produit préférera planifier un projet par la livraison de fonctionnalités et mesurera le succès d'un projet en fonction de la couverture des besoins affaires d'une solution logicielle. Parce que le responsable de produit et le chargé de projet ne considèrent pas les mêmes facteurs de planification, il est normal de penser que les indicateurs de suivi seront différents et que la communication de l'état d'avancement devra être adaptée en conséquence.

6.3.2 Transformation du rôle de chargé de projet[1]

Les approches agiles, en particulier la méthode Scrum, remettent en question le rôle du chargé de projet. Les responsabilités normalement attribuées aux chargés de projet se partagent entre plusieurs autres intervenants :

- la coordination et l'affectation des tâches sont assurées par les équipes autogérées ;
- la gestion, la planification et le suivi sont assurés par le responsable de produit ;
- la productivité, l'efficacité et la collaboration sont des enjeux de l'équipe et celle-ci reçoit l'appui d'un coach l'accompagnant dans cette démarche.

Le partage de responsabilité du chargé de projet vers le responsable de produit n'est pas toujours bien accueilli. Pour plusieurs, le chargé de projet est un rôle d'autorité et il est le principal individu responsable de la réussite d'un projet. Plusieurs organisations et chargés de produit, même s'ils désirent adopter l'agilité, acceptent mal cette transformation. Pour diminuer la possible réticence au changement, il faut que l'organisation soit convaincue de la pertinence du partage de responsabilité pour ensuite travailler avec chaque chargé de projet impacté afin d'identifier avec lui comment son rôle sera transformé.

Ensuite, l'organisation doit prendre conscience qu'elle ne peut pas retirer l'autorité aux chargés de projet et du même coup maintenir la même forme d'imputabilité envers eux. L'organisation doit donc être prête à modifier ses attentes envers les chargés de projets, les responsables de produit et les équipes de projet.

1. Ce rôle peut porter différents titres selon l'organisation : gestionnaire de projet, responsable de projet, etc. Nous parlons ici du rôle responsable de mener les principales activités liées à la gestion de projet (démarrage, planification, exécution, contrôle et suivi, et fermeture).

La transformation des individus

Selon la spécialité de chacun des chargés de projet, certains individus se transformeront dans l'un des rôles prévus par les approches agiles. Certains retourneront vers leur ancienne passion et rejoindront l'équipe de développement. D'autres, ayant acquis une expertise du domaine d'affaires, seront amenés à devenir responsable de produit. D'autres, remarquant des similitudes avec le rôle du chargé de projet, seront tentés de devenir Scrum Master.

Le point intéressant entre le rôle de Scrum Master et celui du chargé de projet, c'est qu'ils ont les mêmes préoccupations. Ce qui les distingue cependant, c'est le mode d'intervention : le Scrum Master n'a pas un rôle d'autorité, c'est plutôt celui d'un facilitateur. Il doit :

- Éliminer les obstacles rencontrés durant le projet, en particulier les problèmes organisationnels dont l'équipe n'a pas la maîtrise.
- Accompagner le client qui n'a pas toujours d'expérience dans le développement logiciel et qui a besoin d'aide au sujet de la planification selon la valeur d'affaires et le retour sur investissement, la gestion de projet et le suivi des incréments de logiciel.
- Aider à la constitution d'une équipe performante, maître d'elle-même, profitant de la collaboration et de l'intelligence collective, capable de s'améliorer et se remettre en question.
- Utiliser les pratiques et la philosophie agiles comme des outils pour relever les dysfonctionnements et proposer des pistes d'amélioration.

Le Scrum Master utilise certaines techniques adaptées à la maturité et la situation de son équipe, comme :

- animer les rencontres pour les garder utiles et efficaces ;
- faire confiance à l'intelligence collective et refuser de croire qu'il contrôle seul la situation ;
- expliquer, convaincre, questionner, montrer les conséquences, laisser l'équipe prendre des risques calculés ;
- faire preuve d'autorité lorsque l'équipe démontre de l'immaturité lui imposant d'être directif, le temps que l'équipe se prenne en main[1].

Certains chargés de projet trouveront la transition difficile, voire impossible. Dans son ouvrage *Succeeding with Agile : Software Development using Scrum*, Mike Cohn propose quelques pistes pour aider cette transition :

1. C'est la théorie du *Situational Leadership*, c'est-à-dire un leadership s'adaptant aux situations, qui a été développée par Paul Hersey, professeur et auteur de l'ouvrage *Situational Leader*, et Ken Blanchard, auteur de l'ouvrage *The One Minute Manager*, pendant la rédaction de l'ouvrage *Management of Organizational Behavior*.

- Respecter les pratiques agiles le plus fidèlement possible, le temps de les comprendre et d'avoir intégré les valeurs sous-jacentes, afin d'éviter de retourner à d'anciennes habitudes.
- Partager le plus possible les retours d'expérience avec d'autres coaches d'équipe, de la même organisation et/ou de la même communauté.
- Lire les nombreuses références disponibles, comme les livres, les articles et les blogues.

La redéfinition du rôle

Certaines organisations en transition agile accordent toujours de la valeur aux chargés de projet. Celles-ci doivent chercher comment ce rôle est modifié et quelle sera la contribution de celui qui le porte au sein des équipes de projet.

Martin Proulx, un collègue agissant à titre de coach organisationnel, a écrit un billet[1] décrivant une démarche pour redéfinir les rôles largement impactés par l'agilité, comme celui du chargé de projet. Il propose de commencer par identifier les responsabilités traditionnellement attribuées au rôle du chargé de projet. Par exemple :

- déterminer les rôles des individus ;
- assigner les tâches aux membres de l'équipe ;
- prioriser les tâches de réalisation ;
- suivre et diffuser les états d'avancement ;
- prendre les décisions du projet ;
- suivre le travail des membres de l'équipe ;
- éduquer et former les employés ;
- gérer les ressources humaines et financières de l'équipe ;
- offrir des plans de carrière et de développement ;
- entretenir les relations avec les différents départements de l'organisation ;
- motiver l'équipe ;
- Diffuser l'information.

Après une analyse de tous les rôles de l'équipe, il est possible d'identifier certaines responsabilités importantes qu'aucun autre individu n'est capable de tenir, soit par manque de ressources ou de compétences. Par exemple, la nouvelle mission d'un chargé de projet pourrait être de :

- définir les objectifs de haut niveau des équipes, plutôt que de se concentrer sur les tâches et les activités de développement ;
- faire le suivi des projets, en mesurant l'atteinte des objectifs ;
- coacher les équipes à la gestion de projet ;
- protéger les ressources humaines et financières de l'équipe ;

1. http://analytical-mind.com/2010/08/12/mommy-i-dont-feel-so-good-im-a-people-manager-in-an-agile-organization/ par Martin Proulx, consulté en mars 2011.

- supporter les individus dans leur plan de carrière ;
- entretenir les relations avec les différents départements de l'organisation.

Rappelons que cette section est un exemple et que le résultat de cet exercice est propre à chaque organisation.

6.3.3 L'adaptation du bureau de projet

Le bureau de projet doit s'assurer que l'organisation obtienne le meilleur rendement sur le capital investi en développement et que les projets soient correctement financés. Comme décrit ultérieurement dans ce chapitre, l'agilité propose de faire le suivi orienté sur le contenu alors que plusieurs gestionnaires préfèrent suivre la consommation des ressources.

Plusieurs praticiens agiles utilisent une technique d'estimation par affinité[1], comme le Poker Planning®. Ce type d'estimation permet d'évaluer la complexité relative d'une exigence en comparaison avec les autres éléments du carnet de produit. La valeur des estimations est indépendante de la capacité de l'équipe. Les estimations relatives de la taille et la complexité étant indépendantes de la vélocité de l'équipe, elles permettent de modifier le carnet de produit et la composition de l'équipe sans nécessiter la réévaluation du carnet de produit.

Un piège que nous observons fréquemment, c'est la conversion des points d'effort des éléments du carnet de produit en jours-personnes. Bien que cette conversion soit rassurante en exprimant le travail restant avec un indicateur compréhensible par les gestionnaires, elle va à l'encontre de l'objectif initial derrière l'utilisation des points de complexité car elle établit une relation fixe entre la taille et la vélocité, ce qui rend difficile la révision du plan de livraison par après (Figure 6.4).

Conseil : Une indépendance entre la taille estimée et la vélocité est souhaitable parce que la taille ne change pas dans le temps, mais que la vélocité de l'équipe change à chacune des itérations même si elle tend à se stabiliser, voire à s'améliorer lorsque les rétrospectives visent l'élimination des problèmes et du gaspillage.

6.4 LA CAPITALISATION

La gestion avec une approche empirique et itérative offre certains avantages, notamment :

- Une définition de terminé complète permettant de mesurer les délais de mise en production et de réduire la durée nécessaire aux activités de la période de stabilisation.

1. Introduit au chapitre *1 Qu'est-ce que l'agilité ?*

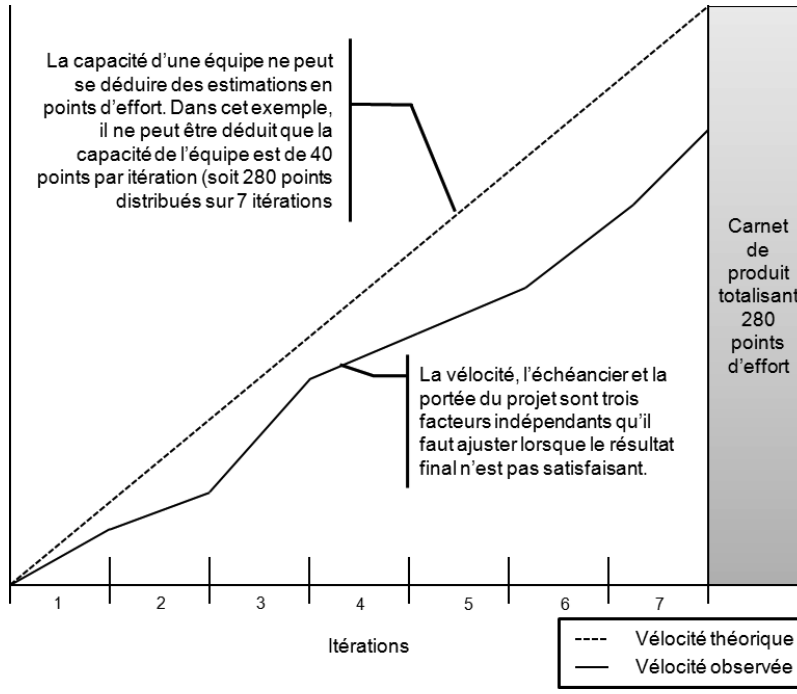


Figure 6.4 — La vélocité ne peut-être imposée car elle représente la vitesse réelle avec laquelle l'équipe livre les exigences réalisées. Elle est l'un des trois facteurs à suivre pour ajuster le plan de livraison.

- Une planification orientée par la valeur d'affaires offrant l'opportunité de capitaliser sur la livraison partielle d'un système en cours de développement.

La capitalisation est un aspect particulièrement intéressant pour ceux qui investissent dans le projet. Prenons l'exemple du développement d'une application de commerce électronique. Avec une planification orientée sur les couloirs fonctionnels[1], il n'est pas nécessaire d'offrir tous les modes de paiement pour permettre la vente de produits en ligne. Le service peut être utilisé avec quelques fonctionnalités d'inventaire, un panier électronique minimal, un seul mode de paiement et un système de facturation et de livraison. La Figure 6.5 illustre comment la capitalisation s'opère avec une approche itérative.

Il est possible que le développement d'un premier couloir fonctionnel demande plus d'une itération de développement. Cet investissement est représenté par les boîtes des *Itérations* où des ressources sont consommées pour livrer de la valeur d'affaires sous la forme d'un nouvel incrément de logiciel. À la Figure 6.28, le système devient utile après l'itération n° 2, où l'option de *Capitalisation* peut être choisie

1. L'utilisation des couloirs fonctionnels pour découper le projet en livraisons est expliquée au chapitre 5 *Équipes et livraison incrémentale*.

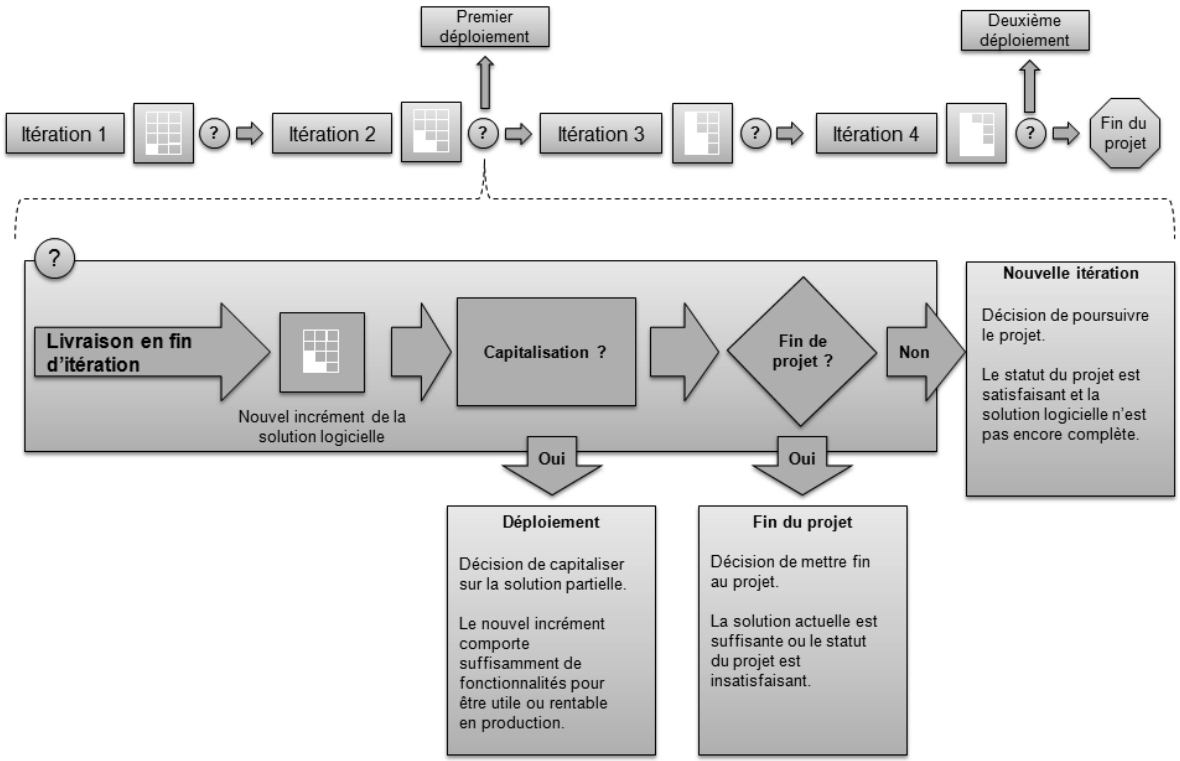


Figure 6.5 — Capitalisation avec développement itératif et incrémental.

pour livrer l'application en production. Par la suite, et à chacune des itérations subséquentes, il peut être rentable d'enrichir l'application en production avec de nouvelles fonctionnalités. Les options de *Fin de projet* illustrent l'alternative de la fin prématurée d'un projet, par exemple lorsque le projet ne donne pas les résultats attendus et qu'il est préférable d'arrêter pour mieux recommencer. Plus positivement, un projet peut également être écourté parce que les fonctionnalités livrées sont suffisantes pour répondre aux besoins d'affaires et qu'il est inutile de continuer à consommer inutilement les ressources allouées.

La capitalisation permet d'éviter le gaspillage des ressources sur un projet voué à l'échec. Elle permet également d'amortir les investissements de développement en générant des revenus à l'aide d'une version partielle du système. Cet amortissement n'est pas toujours monétaire.

6.4.1 La période de stabilisation

Pour que la capitalisation soit possible, il est primordial que l'équipe de projet s'engage à livrer un incrément de logiciel de qualité production à chacune des itérations. Autrement, la dette technique accumulée[1] devra être absorbée avant de pouvoir livrer l'application.

L'organisation doit donner les moyens aux équipes de projets de livrer fréquemment et fidèlement le résultat de leurs itérations sur un environnement le plus représentatif de celui de la production, afin de prévenir l'accumulation d'une dette technique. La gestion de cette dette réduit le temps nécessaire à la stabilisation. Une organisation accordant de l'importance à la qualité de chacun de ses incréments logiciels démontre un état d'avancement de ses projets fiable et constant, permettant de faire de meilleures projections à long terme. Une telle organisation peut s'engager à livrer ses produits sur des périodes de stabilisation aussi courtes qu'une semaine, voire moins d'une journée.

> Commentaire d'un détracteur : « *Notre définition de terminé inclut la validation par un groupe d'utilisateurs finaux et la rédaction d'un guide utilisateur. Est-ce que le respect de la définition de terminé implique de mettre à jour la documentation et faire des tests utilisateurs à chacune des itérations ? Cela me semble trop contraignant.* »

En réponse à ce détracteur, toutes les étapes nécessaires pour qu'un projet soit livré en production devraient être réalisées le plus fréquemment et le plus tôt possible. Lorsqu'une l'organisation préfère repousser certains éléments obligatoires, elle décide alors d'allonger la période de stabilisation précédant les mises en production. Cela a un impact négatif sur la valeur acquise puisqu'après la livraison de la valeur d'affaires, la dette technique fait en sorte qu'il reste de l'effort à dépenser pour tout compléter.

1. Voir le chapitre 5 *Équipes et livraison incrémentale* pour plus de détails sur la dette technique et les façons de l'éliminer ou de la maintenir à un niveau ayant peu d'impact sur la durée de la période de stabilisation.

Conseil : Il est prudent de vérifier de temps à autre l'état et la valeur de la dette technique d'un projet, soit la quantité de travail restant après avoir livré des fonctionnalités et caractéristiques, pour s'assurer de la qualité des indicateurs de gestion et permettre de les ajuster au besoin. Une vérification périodique est une bonne pratique permettant d'enrichir la définition de terminé et d'ajouter des éléments au carnet de produit, rendant ainsi visibles les éléments constituant la dette technique.

6.5 LA GESTION DES RISQUES

Dans un contexte de gestion avec un processus traditionnel, certaines hypothèses sont posées :

- Le risque se trouve dans le plan d'exécution : alors le chargé de projet accorde beaucoup d'importance à ce plan.
- Le risque se trouve dans les spécifications : le chargé de projet émet les hypothèses que les spécifications sont conformes aux besoins d'affaires et que la personne qui estime l'effort de réalisation le fait de façon juste, même si elle ne participe pas au développement.

Cependant, tant que les spécifications sont théoriques, c'est-à-dire qu'elles ne sont pas encore développées et que le résultat n'est pas validé par le client, le projet accumule une dette autour de la faisabilité et de l'acceptation du produit. Le risque survient alors relativement tard dans le projet.

Les approches agiles impliquent des mécanismes d'inspection et d'adaptation. En livrant tôt et fréquemment, elles permettent de vérifier les hypothèses de la gestion de risque beaucoup plus tôt à travers les fonctionnalités terminées d'une itération à l'autre. Par contre, les approches agiles donnent moins de visibilité à long terme sur la durée d'un projet. Les seules certitudes sont les fonctionnalités terminées à ce jour, laissant le risque dans les items restant à faire. C'est pourquoi il est préférable que la priorisation des fonctionnalités à développer tienne compte des risques associés à leur développement, couplés à leur valeur d'affaires et au coût de développement estimé par l'équipe.

6.5.1 Budgéter l'atténuation et la contingence des risques

Dans un contexte non agile, il est fréquent de rencontrer des organisations qui budgètent un pourcentage fixe et linéaire pour atténuer les risques des projets ou en traiter la contingence, par exemple de 20 % ou 30 %. C'est ce qu'on appelle le « *budget de contingence* ». La valeur attribuée est souvent issue de données historiques. Quand un risque est identifié, un ensemble des tâches requises est créé à l'intérieur du plan pour atténuer ce risque. L'effort requis pour ces tâches est alors soustrait du budget de contingence. Il en est de même quand un risque se produit et qu'une série d'activités doivent être déployées pour contingenter l'effet du risque. Quand le budget

de contingence est épuisé mais insuffisant, le chargé de projet a plusieurs choix : demander du budget supplémentaire, couper dans les fonctionnalités à livrer, reporter la date de livraison ou accepter une dette relative à la qualité.

Dans les projets agiles, l'approche des risques se fait autrement : à chacune des itérations, les équipes agiles tiennent compte des risques directement dans l'estimation de chacun des items du carnet de produit. Ces risques sont discutés avec le responsable de produit de sorte que les items plus risqués deviennent aussi plus prioritaires quand leur valeur d'affaires est élevée. En développant ces items risqués plus tôt dans le cycle de vie du projet, le niveau de risque diminue rapidement, laissant peu de chances aux mauvaises surprises de fin de projet.

6.6 LES LIVRABLES

L'utilisation de nouveaux indicateurs peut demander de modifier des documents de suivi. Le bureau de projet, responsable du suivi de projet, doit donc les adapter en conséquence. Dans cette section, nous proposons différents gabarits de livrables associés à la gestion de projet en mode agile que nous avons eu l'occasion d'utiliser dans le cadre de nos interventions.

6.6.1 Le bilan d'itération

Le bilan d'itération (Figure 6.6 et Figure 6.7) est un outil permettant de diffuser, au terme de chaque itération, l'état d'un projet.

Pourquoi ?

- pour communiquer les réalisations et les faits marquants de l'itération ;
- pour conserver l'historique du projet ;
- pour alimenter les discussions du comité de produit ;
- pour informer le bureau de projet.

Comment ?

L'intention est d'inclure l'information la plus pertinente et éviter de devoir transformer cette information pour qu'elle devienne utile au reste de l'organisation. Par exemple, un bilan pourrait contenir :

- le plan de livraison, sous la forme d'un graphique et d'un échéancier ;
- le graphique d'avancement de l'itération ;
- les faits saillants ;
- le carnet de risques et d'obstacles.

Combien ?

Un bilan d'itération devrait pouvoir tenir sur une ou deux pages.

Exemple

Projet ABC

Faire la maintenance perfective du parc applicatif du département des demandes de réclamations d'assurance aux particuliers.

Les objectifs sont :
- La réduction du temps de traitement des demandes de 15 %;
- La diminution du temps d'intégration des conseillers en réclamation.

Faits marquants

- Plusieurs éléments de l'engagement de l'itération 3 n'ont pas été livrés dû à une sous-évaluation;
- Le serveur de pré-production est prêt;
- Un nouveau développeur a rejoint l'équipe.

Plan d'amélioration

Symptôme :
Les éléments de 13 points et plus sont risqués et l'équipe n'est pas en mesure de s'engager à les livrer en 3 semaines.

Proposition :
Révision des règles d'engagement :
- Découpage systématique des éléments de 13 points et plus;
- Planifier des périodes de preuves de concept lorsqu'un élément semble imprécis ou indivisible;
- Découper les tâches de manière à travailler à plusieurs sur un même élément.

Risques

L'équipe doit porter une attention particulière à terminer le travail incomplet de l'itération 3 avant d'entreprendre de nouvelles Itérations. Les scénarios ABC-11 et ABC-16 sont nécessaires à la livraison partielle planifiée à la fin de l'itération 5.

Historique l'itération 3

Story	Poids	Statut
ABC-6	3 points	Terminé
ABC-10	5 points	Terminé
ABC-11	13 points	Incomplet
ABC- 16	13 points	Incomplet

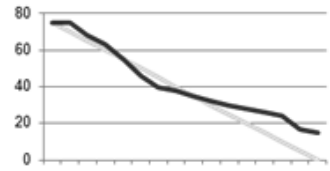


Figure 6.6 — Exemple d'un bilan d'itération (page 1 de 2).

Planification de l'itération 4

Story	Poids	Description
ABC-16	13 points	En tant que conseiller, je veux ....
ABC-11	13 points	En tant qu'assuré, je veux ....
ABC-20	3 points	En tant que conseiller, je veux ....

Plan de livraison

Itération	Début	Fin	Vélocité	Notes
1	07/07/2010	27/07/2010	10	
2	28/07/2010	17/08/2010	7	
3	18/08/2010	07/09/2010	13	
4	08/09/2010	28/09/2010		
5	29/09/2010	19/10/2010		Livraison 1
6	20/10/2010	09/11/2010		
7	10/11/2010	01/12/2010		Livraison 2

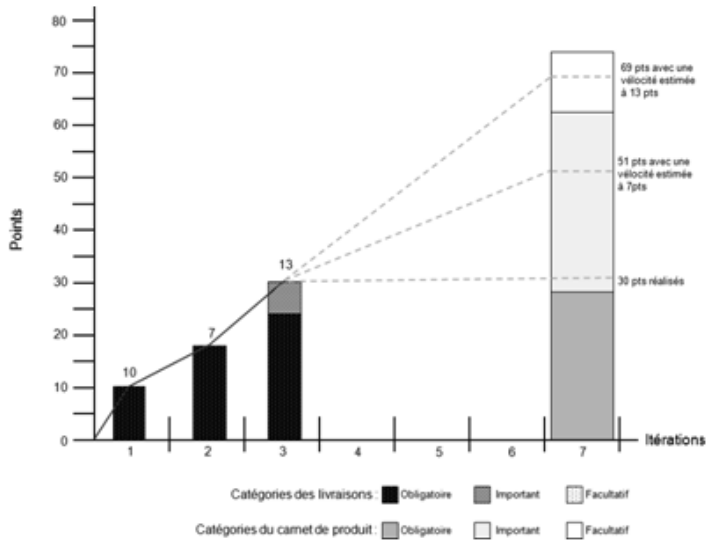


Figure 6.7 — Exemple d'un bilan d'itération (page 2 de 2).

6.6.2 Le plan de livraison

Le plan de livraison documente l'état d'avancement d'un produit et la planification projetée du projet.

Pourquoi ?

Dans son livre *Agile Estimating and Planning*, Mike Cohn énumère trois fonctions du plan de livraison :

- mesurer la progression du projet et identifier le travail restant ;
- afficher l'historique et le fournir à une équipe qui s'engage dans une nouvelle itération ;
- fixer les attentes des parties prenantes du produit.

Comment ?

Un plan de livraison peut facilement se limiter à un échéancier montrant les jalons importants jumelé avec un graphique d'avancement, comme illustré dans l'exemple de la Figure 6.8.

Le graphique d'avancement peut prendre plusieurs formes, selon les aspects à mettre en évidence. Nous proposons trois différents modèles à titre d'exemple :

- l'histogramme des livraisons restantes[1], tiré de l'ouvrage Agile Estimating and Planning de Mike Cohn ;
- le diagramme de stationnement[2], tiré du site web http://leadinganswers.typepad.com[3];
- le sunset graph, conçu dans le cadre de nos interventions.

Histogramme des livraisons restantes

Ce graphique permet de présenter à la fois le travail restant, le travail accompli à chacune des itérations et l'historique des mouvements du carnet de produit. Voici les quatre règles permettant de tracer le graphique :

1. Lorsqu'une demande est complétée, la colonne est réduite par le haut ;
2. Lorsqu'une demande est réévaluée, la colonne s'allonge ou se réduit par le haut, selon la nouvelle valeur ;
3. Lorsqu'une demande est ajoutée, la colonne s'allonge par le bas ;
4. Lorsqu'une demande est retirée, la colonne est réduite par le bas.

La Figure 6.8 illustre le résultat obtenu par l'application des différentes règles.

1. Traduction libre de « *Release Burndown Bar Chart* ».
2. Traduction libre de « *Parking lot chart* ».
3. http://leadinganswers.typepad.com/leading_answers/2007/02/summarizing_pro.html

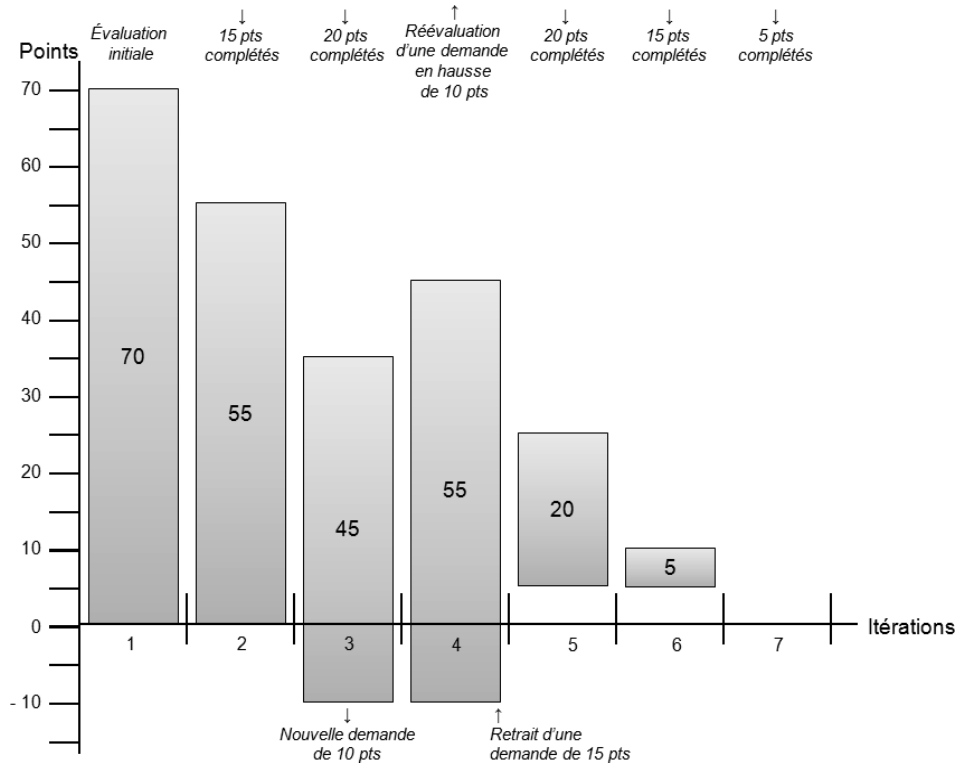


Figure 6.8 — Exemple de l'utilisation d'un histogramme des livraisons restantes.

Diagramme de stationnement

Il est parfois difficile de faire le suivi des thèmes composant un carnet de produit parce que l'ordonnancement par priorité mélange les éléments de plusieurs thèmes. Un thème est mis en œuvre par un certain nombre de cas d'utilisation et chacun de ces cas d'utilisation peut être découpé en plusieurs scénarios utilisateur (*user stories*). Le diagramme du stationnement, qui tire son nom de la forme d'une place de stationnement d'une voiture, permet de mesurer l'état d'avancement d'un projet par thème et par cas d'utilisation.

À la Figure 6.9, le graphique permet de consulter rapidement le nombre et le poids total des éléments d'un cas d'utilisation ainsi que la progression de son développement.

L'exemple de la Figure 6.10 démontre comment il est possible de cartographier les différents thèmes d'un projet et d'afficher globalement l'état d'avancement de chacun d'entre eux.

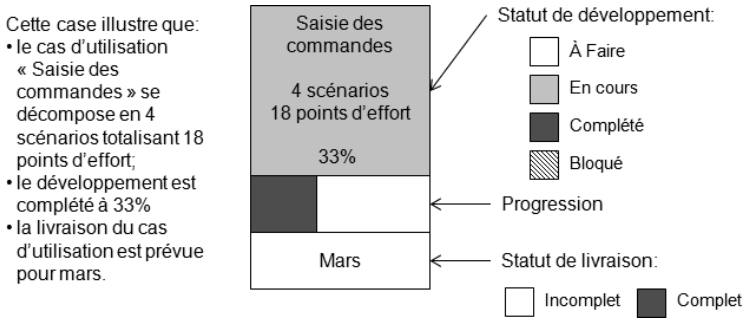


Figure 6.9 — Exemple d'un thème sous la forme d'un diagramme de stationnement.

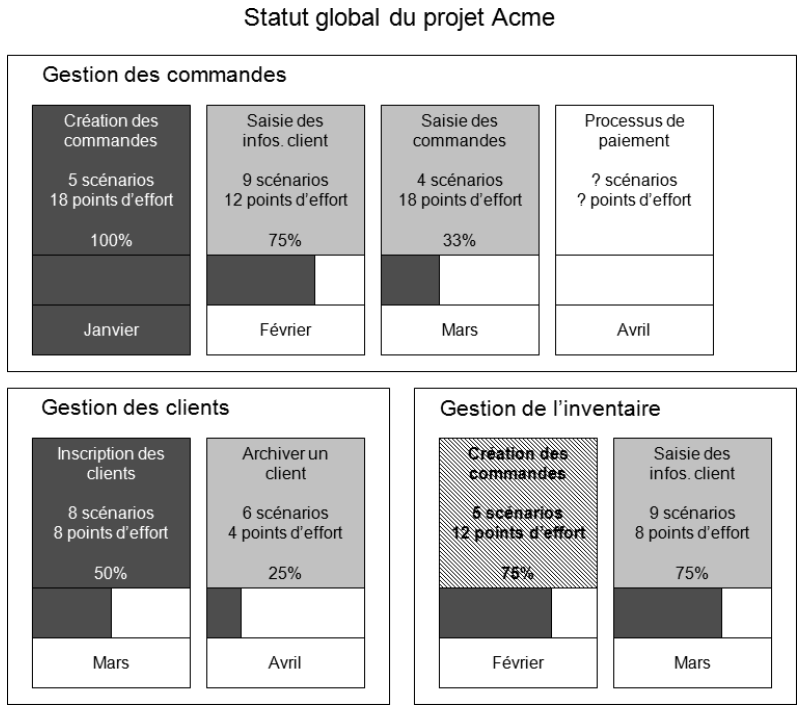


Figure 6.10 — Exemple d'utilisation du diagramme de stationnement pour un ensemble de thèmes.

Sunset graph

Le *sunset graph*[1] a été conçu par Mathieu Boisvert, un des auteurs du présent ouvrage, et Elsa Larreur, une conseillère experte dans les méthodes agiles, dans le cadre d'une

1. Le graphique tire son nom des reflets dégradés d'un coucher de soleil, où chacune des teintes représente la quantité de points d'efforts répartis dans les catégories d'importance de chaque élément du carnet de produit.

intervention en clientèle. Les concepteurs du graphique voulaient exprimer que le succès d'un produit ne dépend pas uniquement de la livraison complète du périmètre des fonctionnalités envisagées au commencement d'un projet. Cet argument est vrai lorsque le produit final répond efficacement au besoin exprimé.

Les intentions du *sunset graph* sont :

- d'illustrer la progression d'un projet en comparaison avec les seuils de satisfaction d'un projet ;
- de découpler la capacité de l'équipe de la portée du carnet de produit ;
- de planifier avec une marge d'incertitude.

La première étape pour construire le graphique est d'identifier les discriminants de satisfaction. Par exemple, à la , les éléments du carnet de produit ont été classés en trois catégories :

- Obligatoire : les fonctionnalités essentielles ;
- Important : les fonctionnalités avec une forte valeur ajoutée ;
- Facultatif : les fonctionnalités attrayantes mais n'ayant pas d'impact sur la qualité fonctionnelle du produit.

À partir de la durée du projet et en estimant la capacité de l'équipe, il est possible d'estimer le nombre de fonctionnalités pouvant être livrées au terme d'un certain nombre d'itérations. Avec des projections pessimistes et optimistes, le seuil de satisfaction du produit final peut être mesuré. Si le seuil estimé n'est pas convenable, il est alors possible de revoir le périmètre du projet, d'apporter des changements à la composition de l'équipe ou de réviser la durée du projet.

D'une itération à l'autre, le graphique trace l'état d'avancement du projet en indiquant la proportion et la nature des fonctionnalités qui sont livrées. La Figure 6.11 indique qu'à la fin de l'itération n° 3, trente points ont été livrés et que la majorité des fonctionnalités obligatoires ont été réalisées.

Au fil du projet, l'origine et l'amplitude de la projection sont ajustées avec l'historique du projet. À la Figure 6.12, la projection pessimiste se base sur les valeurs limites de la vélocité observée : la plus basse étant de 7 points et la plus élevée de 13 points. Le graphique illustre le fait que, si la tendance se maintient, la livraison complète du système n'est pas envisageable :

- il manquera quelques items facultatifs ;
- il risque de manquer certains items importants ;
- la livraison des items obligatoires n'est pas compromise.

Pour s'assurer du succès du projet, il faut vérifier que les éléments situés sous la barre du cas pessimiste sont bien essentiels et qu'il est effectivement possible de se priver des éléments au-dessus du seuil optimiste. Pendant le déroulement du projet, on peut s'attendre à ce que l'angle du cône d'incertitude se réduise : parce que la vélocité de l'équipe est de plus en plus stable et parce que l'origine du cône se déplace vers la

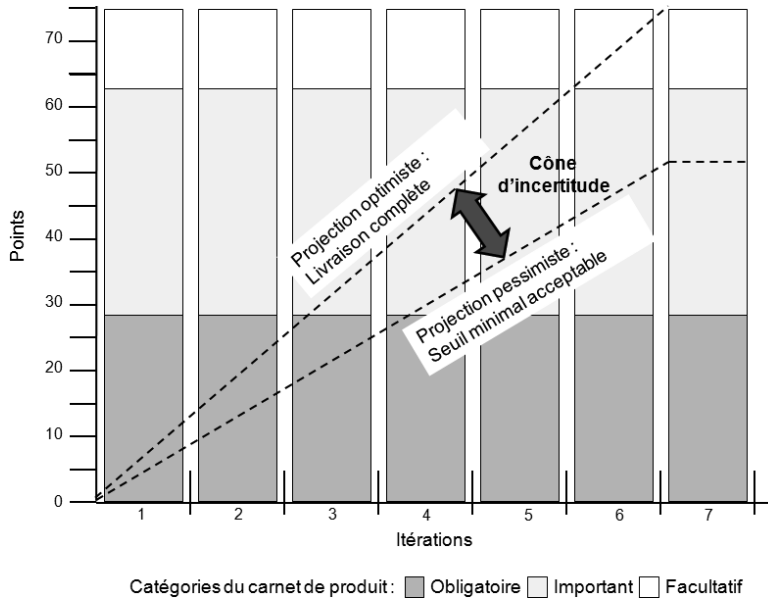


Figure 6.11 — Exemple d'un *Sunset graph* au commencement d'un projet.

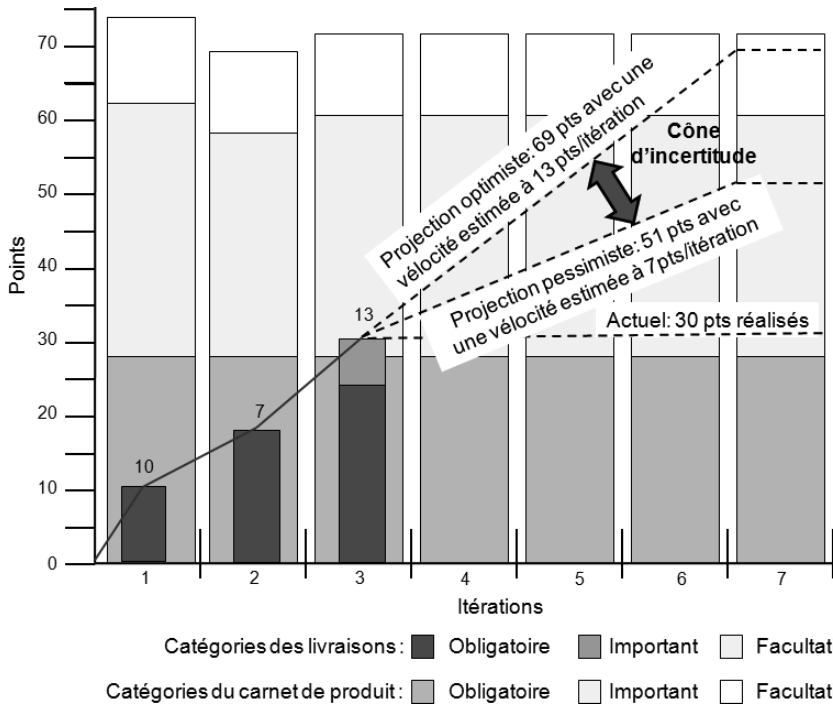


Figure 6.12 — Exemple de l'état d'avancement pendant le déroulement du projet.

droite à chaque itération. Par exemple, le cône d'incertitude présenté à la Figure 6.13 est plus fiable est précis que celui des précédents graphiques.

La Figure 6.13 illustre qu'il est possible de conserver l'historique des changements du carnet de commandes, à l'exemple de l'histogramme des livraisons restantes. Les colonnes du fond représentent l'historique de l'état du carnet de produit pour chacune des itérations.

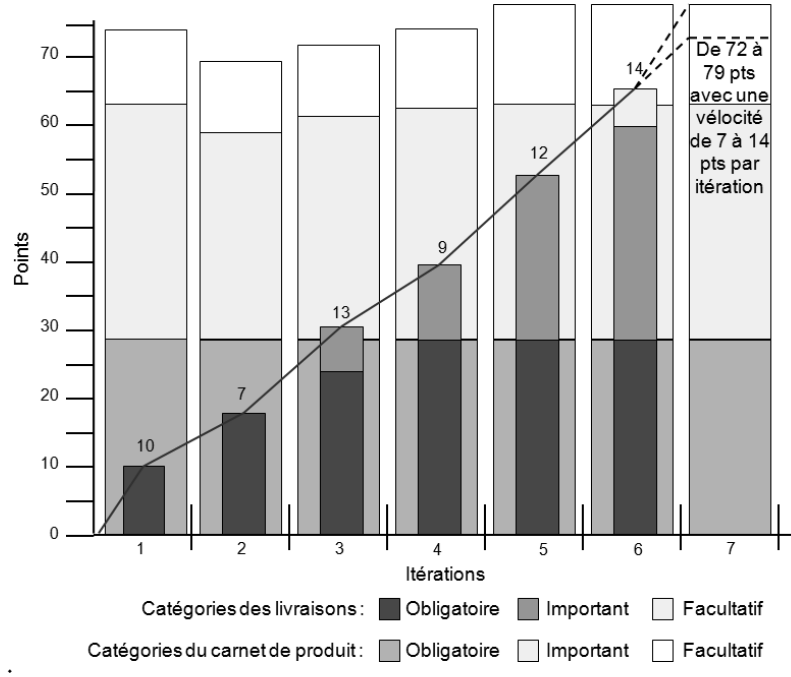


Figure 6.13 — Exemple de l'état d'avancement du projet

Combien ?

Un plan de livraison concis, sous la forme d'un tableau et d'un graphique est très évocateur, peut tenir l'intérieur du bilan d'itération, soit moins d'une demi-page.

Conseil : Bien qu'il existe une corrélation entre la consommation du budget et la livraison de valeur ajoutée, les deux facteurs sont difficilement représentables sur un même graphique. C'est en partie dû au fait que la tendance de la livraison est plutôt logarithmique tandis que la consommation du budget est un facteur plutôt linéaire. Dans un billet du site *Analytical Mind*, l'auteur Martin Proulx nous démontre comment il utilise le *sunset graph* pour faire le suivi de projet et comment il affiche conjointement mais séparément la consommation budgétaire (Figure 6.14).

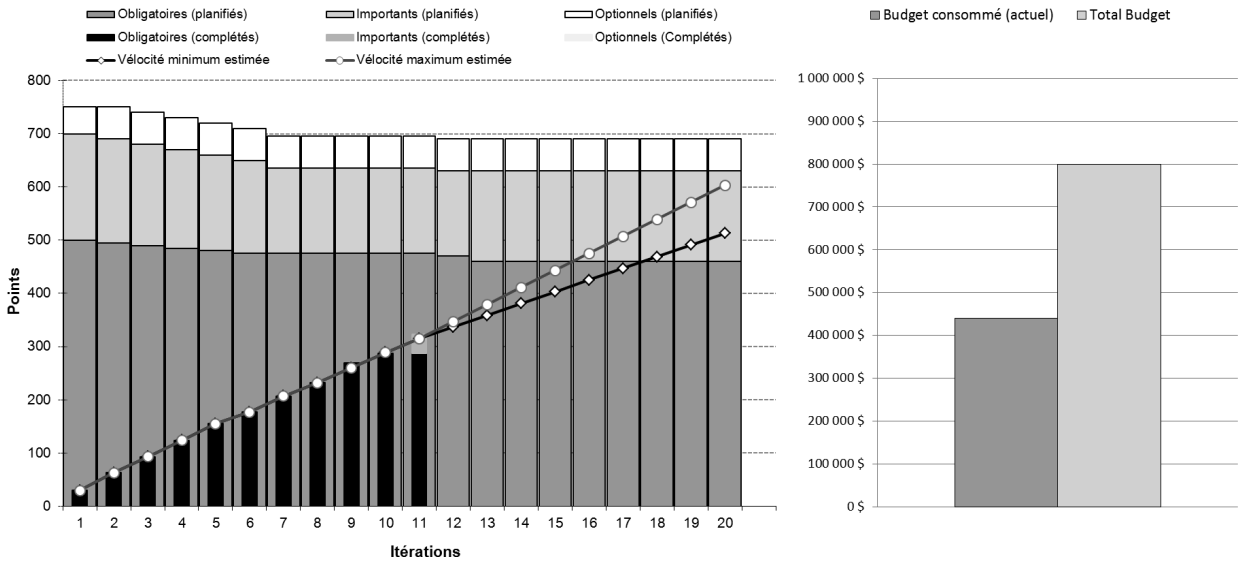


Figure 6.14 — Plan de livraison sous la forme d'un *Sunset Graph* jumelé à un suivi de budget.[a]

a. Martin Proulx, *Status reporting in an Agile context – Introducing the Sunset Graph*, http://analytical-mind.com/2010/12/07/status-reporting-in-an-agile-context-%E2%80%93-welcome-to-the-sunset-graph/, consulté en janvier 2011.

En résumé

Nous avons vu la différence entre une approche déterministe et une approche empirique. Avec son modèle itératif et d'introspection, les approches agiles sont davantage empiriques que déterministes, d'où l'importance de planifier 'juste à temps' et de réajuster cette planification détaillée au gré des itérations.

Le budget, les délais et la portée sont les trois leviers d'ajustement de la gestion d'un projet de développement logiciel. Dans un projet orienté contenu, comme les approches incrémentales, le budget et les délais ne sont plus des mesures de progression, mais plutôt des contraintes de gestion.

Une approche itérative donne des points de visibilité qui permettent à la fois de gérer le risque d'un projet et de capitaliser sur le produit livré de manière incrémentale.

À défaut d'autres intervenants, c'est le client de l'équipe de projet qui est responsable de la gestion de projet, mais dans de grandes organisations, il doit collaborer avec le chargé de projet et le bureau de projet. Il est important de comprendre comment ce nouveau mode de fonctionnement transforme le rôle du chargé de projet, dont les responsabilités sont largement distribuées entre l'équipe et le client.

Finalement, il existe quelques manières différentes de construire des états d'avancement n'utilisant pas le budget comme mesure de progression, et pouvant tenir dans un court bilan d'itération.

7

Déploiement

Objectif

Dans ce chapitre, vous constaterez comment les approches agiles peuvent contribuer et s'adapter aux différents contextes de déploiement des projets de développement.

Mise en situation : Marcel, responsable de la formation

Marcel est responsable de la formation d'un système de comptabilité pour les petites et moyennes entreprises. Depuis que son organisation développe avec des approches agiles, les responsables de produit lui mettent de plus en plus de pression pour qu'il prépare son matériel et son équipe de formation avant la fin du projet, de manière à devancer les dates de mise en service.

Marcel : *Comment puis-je préparer mon matériel si l'application n'est pas complétée ?*

Claude, un responsable de produit : *Pourquoi ne prépares-tu pas ton matériel à l'avance ? Tu pourrais consulter nos livraisons à la fin de chaque itération.*

Marcel : *Je veux bien, mais je n'ai pas encore compris comment en profiter. Vos écrans changent d'itérations en itérations, ce qui me demande de modifier continuellement les captures d'écrans de ma documentation.*

De toute évidence, bien que Claude perçoive un avantage à la livraison incrémentale, Marcel ne comprend toujours pas comment son travail pourrait en tirer avantage. Claude devra comprendre le contexte de Marcel pour être en mesure de l'aider à adapter ses pratiques de travail.

(À suivre...)

7.1 LA PORTÉE DES ACTIVITÉS DE DÉPLOIEMENT ET DE LIVRAISON

Dans cet ouvrage, nous faisons la distinction entre le produit logiciel et les processus d'affaires qu'il supporte tous deux contenus dans la solution d'affaires (Figure 7.1). Chacun d'eux peut avoir son propre cycle de déploiement dicté par des besoins distincts. Prenons l'exemple d'un système de gestion des réclamations d'assurance. Dans ce domaine, nous avons les éléments suivants :

- un **produit logiciel** : l'application de gestion des réclamations d'assurance ;
- un **processus d'affaires** : les flux de travail incluant les rôles et responsabilités et comprenant les procédures opérationnelles, avec ou sans l'utilisation du produit logiciel (certaines procédures peuvent être manuelles, comme de ranger des pièces justificatives dans un classeur) ;
- une **solution d'affaires** : le personnel formé exécutant le processus d'affaires et utilisant le produit logiciel, ainsi que les clients pour lesquels une valeur d'affaires est livrée.

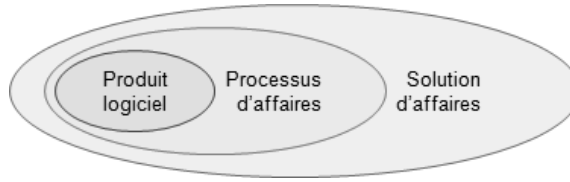


Figure 7.1 — Le développement et le processus sont des sous-ensembles d'une solution d'affaires.

Dans ce contexte, nous distinguons « livraison » et « déploiement » comme suit :

- **Livraison** : installer une version d'un produit logiciel sur un environnement spécifique tel que celui des tests intégrés, des tests d'acceptation ou de production. La livraison d'un produit logiciel comprend :
  - la préparation d'une version exécutable ;
  - l'installation de la version sur un environnement donné ;
  - la mise à jour de la configuration ;
  - la mise à jour, au besoin, des données et de leur structure.
- **Déploiement** : mettre en œuvre une nouvelle version du processus d'affaires ou de la solution d'affaires, incluant une nouvelle version du produit logiciel les supportant. Le déploiement d'une solution d'affaires comprend des activités complémentaires, telles que :
  - la formation et la gestion du changement **;**
  - un agenda et une procédure de mise en service ;
  - un plan de communication et de promotion.

Toutes les activités du déploiement ont des degrés d'importance variés et une ampleur différente selon le projet. Ce sont cependant toutes des activités à considérer pour rendre une solution disponible et préparer les utilisateurs à se l'approprier.

7.2 LES ENJEUX ET LES CONTRAINTES DU DÉPLOIEMENT

La fréquence et le nombre de déploiements d'une solution d'affaires dépendent de plusieurs facteurs. Parfois la fréquence est fixée par des objectifs stratégiques comme :

- La **capitalisation :** une organisation gagne à réduire son cycle de déploiement lorsque cela lui permet de faire des profits plus rapidement et cela, même avec une version partielle de la solution d'affaires.
- La **disponibilité des utilisateurs :** il existe des moments propices pour déployer une solution logicielle. Certaines périodes de pointe, comme les vacances ou la fin d'année fiscale, ont une incidence énorme sur la disponibilité des utilisateurs pour intégrer une nouvelle version de solutions d'affaires. Si un déploiement a lieu durant une période d'indisponibilité, il y a de fortes chances qu'il soit mal accueilli.
- Les **dates d'entrée en vigueur :** les dates butoirs de conformité, comme la date effective des lois, normes et règlements ou une date de mise en marché, imposent la contrainte que le déploiement se termine à un moment bien précis.
- La **gestion du risque** : une solution d'affaires n'est véritablement en œuvre que lorsqu'un déploiement est fait et finalisé. L'augmentation de la fréquence de déploiement réduit l'effet tunnel[1] et les problèmes potentiels pouvant survenir pendant la période de stabilisation.
- Le **juste-à-temps** : la livraison rapide des demandes d'amélioration est un avantage compétitif pour une organisation.

D'autres fois, ce sont les contraintes de l'organisation qui influeront sur la fréquence des déploiements :

- La **capacité et la disponibilité de l'équipe de déploiement** : elle pourrait être différente de celle de l'équipe de développement. Par exemple, il est possible que l'équipe de déploiement soit responsable d'autres applications dont la mise en service est prévue en même temps.
- La **durée nécessaire au déploiement** : la durée requise pour effectuer un déploiement est très variable d'une organisation à l'autre, souvent en fonction du nombre d'utilisateurs impactés et de leur dispersion géographique. Mais cette

1. Rappel : l'effet tunnel, décrit au chapitre 3 *Démarrage de projet agile*, survient lorsque la visibilité sur le déroulement du projet se perd pendant plusieurs semaines, voire des mois. C'est lors du déploiement que les utilisateurs découvrent le produit logiciel que l'équipe leur a livré. L'effet tunnel donne souvent un produit inutile, mal ciblé, ne répondant pas aux besoins du client.

durée est aussi influencée par le temps requis pour livrer en production une version du produit logiciel.

Ces facteurs expliquent qu'il n'existe pas de plan de déploiement unique. Par exemple, une organisation pourrait déployer après seulement trois itérations tandis qu'une autre ne déploierait qu'au terme de toutes les itérations du projet (Figure 7.2).

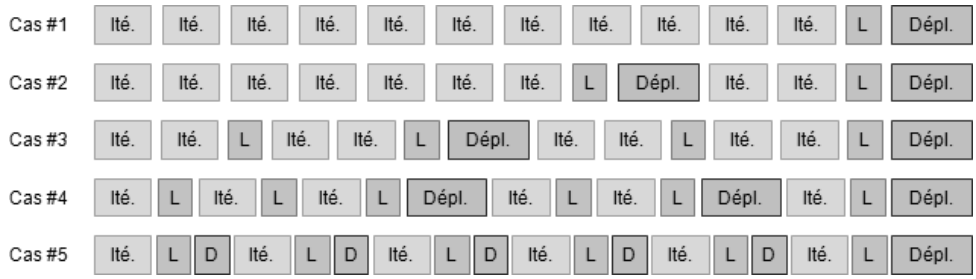


Figure 7.2 — Exemples de fréquences de livraison et de déploiement pour différents contextes de projets.

Malheureusement, ces contraintes peuvent retenir la capacité d'une organisation à déployer, même lorsque le déploiement est basé sur des objectifs stratégiques.

7.3 LES APPROCHES AGILES AU SERVICE DES ENJEUX DU DÉPLOIEMENT

Certaines approches et pratiques agiles permettent de répondre à des objectifs stratégiques et diminuer les contraintes associées aux activités de déploiement. En voici quelques exemples.

7.3.1 L'intégration continue et la livraison automatisée

Le temps nécessaire pour « construire »[1] et livrer un produit logiciel sur un environnement devrait être un facteur **négligeable** des activités de déploiement. En effet, il est dommage qu'une durée de livraison trop longue soit un facteur suffisamment important pour être la raison de la réduction de la fréquence de livraison, au détriment des objectifs stratégiques.

L'intégration continue, l'une des pratiques de la méthode *eXtreme Programming*, propose de construire et livrer les produits logiciels de manière automatique à **chaque soumission** de code au système de contrôle de versions[2].

1. Traduction de l'anglais « build », qui consiste à compiler et assembler une solution logicielle prête à la livraison à partir du code source.
2. Traduction libre de l'expression anglaise : *commit to source version control system*.

Pour que l'intégration continue soit efficace, il faut qu'une livraison puisse se faire avec un outil automatisé, que cet outil s'exécute en moins de dix minutes et que la livraison sur un environnement cible tienne en trente minutes. Avec des performances de ce genre, la livraison quotidienne et automatique par un serveur d'intégration devient possible et l'effort de livraison devient effectivement un facteur négligeable des activités de déploiement. Il faut prévoir du temps d'entretien pour maintenir les outils d'intégration continue et de livraison afin de s'assurer que la vitesse de déploiement demeure un avantage compétitif et que le processus de livraison ne s'étire pas inutilement à cause de solutions de contournement.

La plupart des langages et technologies d'aujourd'hui supportent cette pratique et il existe un bon nombre d'outils et de serveurs de compilation pour adopter la pratique de l'intégration continue. La pratique assure qu'une version fonctionnelle du produit logiciel est prête à être livrée et déployée à tout moment.

Retour d'expérience : réduction des délais de livraison

Dans une grande société financière, le délai requis pour installer une version d'une application sur un environnement de production était de quatre semaines. Leur processus demandait de produire cinq documents pour chacune des livraisons, tous créés à la main dans des fichiers textes ou des tableurs.

Ces délais empêchaient les déploiements fréquents en production.

Avec l'adoption de l'agilité et les rétrospectives visant à améliorer l'efficacité, les équipes ont automatisé la majorité des tests et leur nouvel outil de gestion des versions permet l'extraction automatique des composantes nouvelles et modifiées, incluant la liste des modifications aux interfaces avec les systèmes externes. Étant donné qu'une partie importante de l'information requise était disponible rapidement avec cet outil, les délais pour livrer sur l'environnement de production ont été réduits à une semaine.

Sylvie

7.3.2 Les livraisons fréquentes

Pour gérer le risque, les approches itératives prévoient la livraison d'un incrément de logiciel fonctionnel à chaque itération, même si chaque incrément ne fait pas l'objet d'un déploiement. Ce qui n'empêche pas de livrer ces incréments sur des environnements intermédiaires au service des autres intervenants des activités de déploiement. Cela offre l'avantage d'offrir la transparence requise pour coordonner les différentes activités de déploiement lorsqu'une date butoir est fixée comme, par exemple, dans le cas de la mise en application d'un nouveau règlement.

La livraison partielle d'une solution est également intéressante pour les activités promotionnelles et/ou dans le cadre d'un programme de mise en marché conjoint, car elle permet d'assurer la promotion et le lancement d'une version fonctionnelle et d'utiliser la réduction du temps de livraison comme un avantage compétitif qui réduit les temps de déploiement.

Retour d'expérience : application inadéquate d'un cycle incrémental

J'ai travaillé dans une équipe de développement en sous-traitance dans le cadre de la réalisation d'une solution communautaire devant partager, vendre et diffuser le contenu musical d'artistes indépendants. Notre client appréciait particulièrement les approches itératives car cela lui permettait de réagir rapidement à toutes les occasions possibles de démontrer le produit aux différentes communautés de l'industrie de la musique.

Malheureusement, cet avantage était un couteau à double tranchant. Le client, désirant continuellement impressionner l'industrie musicale, planifiait seulement en fonction de la promotion, négligeant l'objectif de réaliser une solution fonctionnelle. À terme, cette stratégie s'est montrée inefficace, car le produit qui devait au départ se développer de manière incrémentale est devenu un énorme prototype.

Dans ce projet, nous avons manqué à l'éducation du client, qui est tombé dans le piège de l'éparpillement, consommant les ressources disponibles en fonction de résultats immédiats.

Mathieu

7.4 LES RÉPERCUSSIONS SUR LA GESTION DU CHANGEMENT ET LA FORMATION

Adopter l'agilité et son cycle itératif et incrémental a des impacts positifs sur les intervenants impliqués dans les activités de déploiement mais externes aux processus de développement logiciel : il ne faut pas hésiter à revoir les processus et leurs outils car les bénéfices escomptés sont en général plus grands que les coûts engendrés par la transformation. Les intervenants externes étant souvent spécialisés dans leur domaine, ils sont les mieux placés pour orchestrer cette transformation, à condition de leur fournir une formation adéquate sur l'implantation de l'agilité dans leur organisation. Ils doivent accepter de changer leur flux de travail et le fait qu'il y aura une certaine quantité de travail à retoucher.

La réalisation des activités associées à la gestion du changement et de la formation est une activité qui tend à s'approcher de l'exécution des activités du déploiement. Principalement parce que le contenu de la solution finale est un préalable important à la préparation des livrables, comme les registres des incidences, les guides de formation, les guides d'utilisation, l'aide en ligne et les plans de formation. Avec une approche incrémentale, il est possible de devancer la préparation du matériel de formation et de la gestion du changement, parce que les équipes de réalisation sont engagées à livrer une version partielle, fonctionnelle et de qualité de la solution logicielle à toutes les itérations.

Certains, comme Marcel dans la mise en situation en début de chapitre, diront qu'il y aura du travail à retoucher. C'est vrai, mais la règle de Pareto s'applique la plupart du temps : 80 % des fonctionnalités changeront peu tandis que 20 % des

fonctionnalités vont évoluer sur plusieurs itérations, nécessitant une adaptation du matériel de formation et de gestion du changement à chaque évolution.

Mise en situation : Marcel, responsable de la formation (suite)

Claude et Marcel ont échangé sur leurs processus, leurs enjeux et leurs préoccupations. Devant l'incompréhension de Marcel sur le cycle itératif de développement, Claude lui a montré le carnet de produit priorisé, quelques-uns des items avec la documentation associée, puis il lui a fait une visite guidée sur l'environnement d'intégration des fonctionnalités développées au cours des trois premières itérations. Claude lui a également montré le carnet de l'itération courante de façon à ce que Marcel sache quelles allaient être les fonctionnalités à déployer en fin d'itération.

En comparant ce qui avait été accompli, ce qui se faisait, et ce qui allait être développé, Marcel a compris comment il pouvait adapter son processus de sorte qu'il puisse faire des livrables de façon incrémentale. Ainsi, le guide d'utilisation, jadis fait d'un bloc où il inventoriait les fonctionnalités et leur usage, se ferait désormais en fonction du résultat de chaque itération. La version de ses livrables devait donc suivre celle du produit logiciel, ce qu'il faisait en mode traditionnel mais seulement une fois par an au lieu de six à dix fois.

Il savait qu'il aurait du travail à refaire, mais la quantité n'était pas aussi importante qu'il l'avait initialement cru. Il acceptait de refaire, ou plutôt de faire évoluer des parties du matériel de formation, car il voyait maintenant les avantages à déployer de plus petits changements plus souvent.

La livraison fréquente permet aux responsables de la formation et de la gestion du changement de préparer leurs activités avant la fin du développement. Exécutées en parallèle, ces activités réduisent le temps séparant la fin du développement de la mise en production de la solution d'affaires et, du même coup, le temps de déploiement global.

Notre expérience démontre toutefois que les travaux de préparation de la formation et de la gestion du changement ne bénéficient pas toujours de cet avantage potentiel : le résultat des premières itérations est habituellement embryonnaire et appelé à être modifié. Le défi est d'identifier le moment où un incrément de logiciel sera suffisamment complet pour servir de matière aux autres intervenants du déploiement. Nous conseillons donc d'évaluer, au terme de chaque itération, si le bon moment est arrivé pour planifier l'intervention des responsables de la formation et de la gestion du changement.

En résumé

Les activités de déploiement ne se limitent pas à la simple livraison de la solution logicielle. Il est important pour les équipes de réalisation d'offrir une procédure qui n'est pas au détriment d'objectifs stratégiques associés au déploiement.

Les intervenants responsables des autres activités associées au déploiement, comme les responsables de la formation, de la gestion du changement, de la mise en marché ou encore de la gestion du contenu, peuvent profiter du processus incrémental pour leurs activités. Ces intervenants devraient être impliqués tôt dans le déroulement du projet, voire au commencement, pour qu'ils soient en mesure d'identifier la manière de profiter des approches agiles et d'identifier le meilleur moment pour démarrer leurs travaux respectifs, en parallèle au développement.

En plus de soutenir les objectifs stratégiques, les activités en parallèle de tous les intervenants associés au déploiement d'une solution logicielle viendront réduire le temps requis séparant la fin du développement de la date de mise en service.

8

Infrastructure technologique

Objectif

Dans ce chapitre, vous constaterez comment le développement itératif introduit de nouveaux besoins en matière d'infrastructure technologique et vous découvrirez les impacts provoqués sur les équipes d'infrastructure, au point de modifier leur processus de gestion des demandes de service.

8.1 LE DÉVELOPPEMENT VERSUS L'INFRASTRUCTURE

La collaboration entre les équipes de développement et les responsables de l'infrastructure technologique est une relation fragile dans la plupart des organisations. Cette fragilité est causée par la nature différente de leurs responsabilités. Cette différence provoque parfois une division entre les deux corps de métier qui n'arrivent pas à comprendre les préoccupations et les contraintes de leurs collègues.

8.1.1 Dans le coin gauche : les développeurs

Les développeurs utilisent des technologies innovantes offrant fréquemment de nouvelles alternatives. Cette innovation constante permet aux équipes de développement de réagir rapidement aux besoins émergents et changeants de leur clientèle.

Plusieurs développeurs ignorent les contraintes et les exigences des systèmes en production. Certains consultants ou jeunes professionnels n'ont jamais participé à la mise en production et à l'entretien d'un système logiciel. Il est difficile pour eux de comprendre les réalités des responsables de l'infrastructure.

8.1.2 Dans le coin droit : les responsables de l'infrastructure

Les responsables de l'infrastructure mettent à disposition des services se conformant à des critères de qualité exigeants comme la sécurité, l'accessibilité, la tenue en charge, le temps de réponse, et tout autre critère exigé par la clientèle de l'organisation. L'intégrité et la disponibilité des services dont ils sont imputables incitent les responsables de l'infrastructure à être prudents quand vient le moment d'intervenir sur les systèmes. À l'exemple des développeurs ayant besoin d'assurance pour remodeler le code en toute confiance, les responsables de l'infrastructure veulent être certains de ne pas introduire de régression à la suite d'une intervention sur les composantes de l'infrastructure.

Lorsque l'équipe d'infrastructure n'intervient qu'à la toute fin du cycle d'un projet, le produit leur est inconnu. Ils ont besoin de temps pour le maîtriser et analyser l'impact de sa livraison dans les différents environnements. Le temps nécessaire à ces travaux repousse d'autant la mise en œuvre des demandes de livraison et de changements. Leur réticence à intervenir sur des systèmes instables, fragiles ou non maîtrisés peut paraître comme un manque de réactivité alors qu'en fait ce sont leurs précautions qui réduisent leur vitesse d'intervention. Ces précautions irritent souvent la clientèle et les développeurs qui aimeraient plus de réactivité de leur part.

8.1.3 Le bras de fer

Les principes agiles ajoutent de la pression sur cette collaboration fragile. C'est particulièrement vrai lorsque les équipes de développement adoptent une approche agile sans consulter les équipes d'infrastructure.

Même si les équipes de développement peuvent momentanément isoler les impacts de l'agilité sur leurs confrères, tôt ou tard les concepts de livraisons fréquentes et des équipes autogérées finiront par mettre les équipes d'infrastructure devant le fait accompli : le mode de collaboration doit s'ajuster à l'agilité.

8.2 S'ADAPTER AUX IMPACTS DE L'AGILITÉ

Les approches agiles ajoutent un stress sur la collaboration entre les deux équipes. Le développement d'incréments complets de logiciel demande de travailler sur plusieurs environnements en parallèle, et ce à toutes les itérations. Ce mode de fonctionnement multiplie les demandes envers l'équipe d'infrastructure mais n'implique pas qu'elle soit préparée à traiter les demandes d'installation à la pièce, ni à réagir rapidement aux multiples demandes de configurations et d'accès.

8.2.1 Les environnements de développement

Les intervenants d'un projet nécessitent plusieurs environnements différents pour être en mesure de travailler sans être impactés par les activités des autres corps de métiers. Par exemple, les environnements suivants sont typiquement utilisés :

- Un environnement **d'intégration continue** : pour y installer un serveur responsable de compiler et exécuter quotidiennement le plan de tests unitaires.
- Un environnement **fonctionnel** : pour séparer les fonctionnalités en cours de développement de celles prêtes à être démontrées. Cela permet au client de vérifier la conformité des éléments livrés sans se soucier des impacts possibles des activités de développement.
- Un environnement **d'acceptation** : un environnement mis à jour à toutes les itérations et semblable à l'environnement de production, permettant à tous les intervenants de faire les tests d'acceptation sur chacun des incréments de logiciel. Pensons notamment aux tests utilisateurs, aux tests de performance et à ceux de la sécurité.
- Un environnement **de production** : pour mettre la solution logicielle à la disposition des utilisateurs au fur et à mesure que des ensembles cohérents de fonctionnalités sont prêts à être utilisés.

Avec l'adoption de l'agilité et pour être en mesure de livrer de la valeur d'affaires à chaque itération, l'organisation doit investir plus tôt dans l'infrastructure de développement et de déploiement. Ainsi, les délais d'intervention sont réduits car tous les environnements doivent être livrés tôt en début de projet. Cela exige que leur processus de livraison des environnements soit efficace et que l'équipe d'infrastructure adopte une approche incrémentale, c'est-à-dire qu'elle doit :

- **Concevoir l'architecture des environnements de façon émergente** : Bien que ce soit une bonne pratique de concevoir une cible complète basée sur le recueil des besoins, une préparation incrémentale permet d'accélérer la mise à disposition des environnements et ainsi offrir une meilleure réactivité.
- **Limiter l'anticipation** : Tout comme pour les travaux des équipes de développement, les travaux d'infrastructure doivent répondre à des besoins d'affaires. L'installation et la configuration de composants ne répondant pas à un besoin d'affaires immédiat risquent d'être remises en question, voire d'être retirées. Cela représente un gaspillage de temps et d'argent pour les responsables de l'infrastructure.

8.2.2 L'interaction

Retour d'expérience : Collaborer avec l'équipe d'infrastructure

Une équipe de développement que j'accompagnais dans le cadre d'un projet pilote de l'adoption de Scrum n'arrivait pas à tenir ses engagements à cause de leur dépendance avec l'équipe d'infrastructure.

Équipe : « *La préparation de l'environnement des tests de charge s'étire. Chaque fois que nous sommes informés que l'environnement est prêt, nous découvrons des nouveaux problèmes et nous devons placer une nouvelle demande de service à l'équipe d'infrastructure. À chaque fois, cela nous place en attente. Ils expriment le même problème de leur côté, parce qu'ils n'ont jamais la certitude d'avoir complété la préparation de l'environnement.* »

Mathieu : « *Il me semble que le problème, c'est que la préparation est découpée entre vos deux équipes. Plutôt que d'enchaîner vos interventions, avez-vous considéré la possibilité de travailler tous ensemble à la préparation de l'environnement et de vous y concentrer jusqu'à ce que l'environnement convienne à toutes les parties* ? »

Ce nouveau mode de fonctionnement a permis de régler le problème de la préparation de l'environnement des tests de charge. La prise de conscience que tous les travaux ne peuvent pas être entrepris en isolation a permis de résoudre l'inefficacité de la situation.

Aujourd'hui, les règles de préparation des demandes à l'intention de l'équipe de l'infrastructure prévoient la planification de travaux collectifs.

Mathieu

Le précédent retour d'expérience met en lumière que les équipes d'infrastructure sont au service du client via les équipes de développement. À l'exemple du client, responsable de la validation et de la spécification de son système, les équipes de développement devraient être invitées à participer à la conception et l'exécution de leurs demandes de service.

Une collaboration active réduit les mauvaises interprétations causées par l'échange de documents, profite de l'intelligence collective des personnes en place et permet à toutes les parties d'être sensibles aux difficultés et aux conditions de succès des autres.

Les responsables d'infrastructures devraient être accueillants lorsqu'un développeur veut s'impliquer et ils devraient inviter les équipes de développement à s'approprier leur solution en collaborant étroitement avec elles.

8.2.3 Les livraisons fréquentes

Pour protéger les environnements d'une organisation, leur accessibilité est habituellement restreinte et contrôlée. Pour chacun des besoins en installation ou en configuration, une demande de service est placée auprès des équipes d'infrastructure, responsables du bon fonctionnement et de l'intégrité des environnements.

Avec une approche itérative, les développeurs sont amenés à livrer fréquemment de nouvelles versions des applications et à installer des modules de manière répétitive. Pour éviter que ce mode de fonctionnement surcharge de travail les équipes d'infrastructure, il faut revoir l'accessibilité des développeurs pour leur donner la latitude requise à leur travail sans compromettre les contraintes de protection des environnements.

Retour d'expérience : Réduire la pression sur l'équipe d'infrastructure

Dans le cadre d'un mandat de Scrum Master au sein d'une entreprise de télécommunication, les demandes fréquentes et urgentes de livraisons ont rapidement irrité nos correspondants des équipes d'infrastructure.

Après avoir expliqué notre mode de fonctionnement et compris les contraintes d'infrastructure imposées par l'organisation, nous avons convenu avec eux d'augmenter notre autonomie par l'utilisation de *scripts*.

Ces *scripts* permettaient à notre équipe d'installer à volonté notre solution logicielle sur l'environnement d'acceptation sans l'intervention de l'équipe d'infrastructure. Les développeurs étaient responsables de l'écriture et l'entretien des *scripts,* puisqu'ils étaient les mieux placés pour décrire l'installation de leur solution logicielle. Avant d'en permettre l'exécution sur l'environnement d'acceptation, l'équipe d'infrastructure révisait le contenu des *scripts* en vérifiant que leur exécution était conforme aux contraintes d'infrastructure.

Cette solution a augmenté notre autonomie et réduit la pression que nous exercions sur l'équipe d'infrastructure.

Mathieu

Les responsables de l'infrastructure sont des spécialistes, et comme tous les autres spécialistes, ils doivent prendre garde à ne pas devenir les goulets d'étranglement de l'organisation. Ils doivent être créatifs pour donner le maximum d'autonomie aux développeurs. En réduisant le nombre de tâches répétitives, ils libéreront plus de temps pour s'investir dans les activités nécessitant réellement leur expertise.

8.2.4 La qualité production

À l'exemple des solutions logicielles qui déçoivent par leur manque de qualité et ne répondent pas aux conditions de succès attendues, les réponses aux demandes de service d'infrastructure sont parfois source de frustration.

Une installation incomplète, l'oubli de l'attribution des droits à un utilisateur, et une régression de la qualité des services, sont quelques exemples des anomalies brisant le lien de confiance envers les équipes d'infrastructure.

Retour d'expérience : Améliorer la qualité des livraisons

Dans le cadre d'un mandat au sein d'une entreprise spécialisée dans le développement de solutions multimédias, j'avais affaire avec une équipe d'infrastructure externe.

Après quelques erreurs d'installation des versions, nous avons réfléchi ensemble à l'amélioration de la procédure de livraison. J'ai pris conscience de la difficulté à déployer une application par une personne étrangère à son développement.

Pour lui permettre de vérifier lui-même la qualité de l'installation, nous avons convenu d'ajouter à la documentation quelques points de contrôle permettant au responsable des livraisons de vérifier l'opération normale pendant et après l'installation.

Cette nouvelle documentation a considérablement réduit le nombre d'appels téléphoniques en urgence que nous avions à chaque livraison.

Mathieu

À l'exemple de la définition de terminé et des conditions de succès, l'équipe d'infrastructure peut exiger de ses clients de connaître les attentes et les tests de validation vérifiant la qualité d'une livraison. Ces tests servent de spécifications et lui permettent de vérifier la qualité de son travail avant d'indiquer que la demande est complétée. Cette vérification en amont prévient les retours inefficaces des clients insatisfaits et la perte de confiance envers les équipes d'infrastructure.

8.3 LES ADMINISTRATEURS DE SYSTÈME : PARTIE PRENANTE DE LA SOLUTION D'AFFAIRES

Les administrateurs de système sont responsables envers les utilisateurs du support et de la disponibilité du portfolio des applications. Cependant, toutes les solutions logicielles ne sont pas simples à supporter. Trop souvent, les logiciels ont été développés sans considération sur leur maintenabilité, en partie parce que plusieurs développeurs n'ont jamais participé à l'entretien d'un système et ne connaissent pas les enjeux associés à cette activité.

Il y a de la valeur à inviter les administrateurs aux activités de démarrage d'un projet de développement. Non seulement cela leur permet de comprendre les enjeux du projet, de s'informer de l'échéancier et du moment où ils interviendront, mais aussi cela permet de recueillir leurs besoins en matière de maintenabilité, notamment :

- une journalisation claire et ciblée pour diagnostiquer la cause des incidents en production ;
- un système tolérant les erreurs, c'est-à-dire capable d'isoler un mauvais traitement sans bloquer la bonne exécution du programme ;
- un outil de configuration limitant la modification interne du système, et par conséquent, l'intervention des développeurs ;
- une solution compatible avec leurs outils d'analyses de charge, d'activités et de performance.

Les besoins et les attentes des administrateurs peuvent enrichir la définition de terminé des équipes de développement soucieuses d'augmenter la maintenabilité de leur système. De plus, cela augmentera l'appropriation et la satisfaction des administrateurs qui seront responsables du maintien du système.

8.4 PRINCIPES DE L'INFRASTRUCTURE AGILE

Certaines équipes d'infrastructure apprécient le contenu du manifeste agile et désirent ne pas simplement s'adapter aux impacts de l'agilité, mais également adopter une approche agile.

Le site *Agile operations*[1] identifie quatre principes s'appliquant la réalité des équipes d'infrastructure : simplicité, communication, collaboration avec les clients et processus.

8.4.1 Simplicité

La complexité est une cause importante des incidents sur les applications et elle impacte le temps de résolution des problèmes. Outre la complexité interne des

1. Scott Wilson, *Agile Operations*, http://agileoperations.net/, consulté en janvier 2011.

logiciels, il existe aussi une complexité inhérente à l'infrastructure. Plus il y a de composantes installées et configurées pour faire fonctionner une application et plus cette infrastructure est complexe. Une des qualités recherchée par les responsables de l'infrastructure devrait être la conception de solutions simples prévenant l'apparition d'incidents et facilitant leur résolution lorsqu'elles sont inévitables.

La complexité d'une installation et de sa configuration peut limiter le nombre de personnes en mesure d'intervenir lorsque survient un incident. Ce qui n'est pas souhaitable. Plus les personnes maîtrisent l'infrastructure liée à une application, plus elles prendront des décisions assurant son intégrité, et plus elles auront l'assurance d'améliorer la solution sans risque d'introduire de la régression dans la qualité des services.

Au moment d'analyser une solution d'infrastructure, les responsables devraient toujours évaluer plus d'une alternative afin de considérer ou favoriser la plus simple. Notre expérience nous a démontré que des impacts significatifs sur les coûts, les délais et la qualité de service peuvent survenir lorsque cette pratique est escamotée.

Comme décrit à la section sur les environnements de développement, il est avantageux de limiter l'anticipation et de ne pas installer un service ou de configurer une composante lorsque le besoin n'a pas encore émergé. Cela diminue les sources possibles d'incidences sur les applications.

8.4.2 Communication

La documentation des systèmes est un facteur clé dans les activités de soutien et d'installation. Comme toute autre documentation agile, elle mérite d'être claire, utile et facile à maintenir. Les documentations exhaustives sont souvent rébarbatives et difficiles à entretenir.

Dans le contexte d'une équipe de soutien, la documentation est essentielle car elle augmente l'autonomie des intervenants à agir sans briser, mettre à risque, ou corrompre le système.

En plus d'une documentation traditionnelle, plusieurs organisations ont mis en place des bases de connaissances pour partager l'historique des résolutions de problèmes. Ces bases de connaissances sont un bon exemple de l'intelligence collective permettant à une personne, face à un problème, de profiter de l'expérience des autres et agir de manière autonome et efficace[1].

8.4.3 Collaboration avec les clients

Valeur agile : La collaboration avec le client davantage que la négociation de contrat.

1. Nous recommandons la lecture du chapitre de la *14 Agilité et documentation* si vous désirez des conseils pour améliorer la forme et le contenu de votre documentation.

L'équipe d'infrastructure est en mesure de participer à plusieurs pratiques agiles, notamment :

- **La mêlée quotidienne** : en particulier lorsque les travaux demandent une collaboration étroite avec les équipes de développement.
- **L'entretien du carnet de produit** : comme décrit précédemment, l'équipe d'infrastructure est une partie prenante qui peut exprimer ses besoins envers le produit développé.
- **La revue d'itération** : comme n'importe quel intervenant au système, il y a de la valeur à constater l'état d'avancement d'un projet.
- **La rétrospective** : à l'occasion, c'est le bon endroit pour discuter avec l'équipe de développement de l'amélioration des échanges et de la collaboration entre les deux corps de métier. Toutefois ne pas oublier que normalement, la rétrospective est réservée aux membres de l'équipe de développement.

Cet investissement en temps permet de prévenir les fausses attentes et perceptions que le reste de l'organisation peut avoir envers les équipes d'infrastructure. Comme expliqué dans la première section du chapitre, les attentes d'efficacité sont grandes et il est fréquent que, malgré leurs efforts, les responsables de l'infrastructure doivent gérer l'insatisfaction de leur clientèle. Cette insatisfaction provient habituellement d'un manque de réactivité ou d'une réponse inadéquate aux demandes de service exprimées par les équipes de développement ou par la clientèle affaires. En particulier lorsqu'une demande de service est refusée par les équipes d'infrastructure qui sont alors accusées d'être inflexibles et de ne pas être au service des utilisateurs. Puis s'enchaîne une négociation avec les clients où les équipes d'infrastructures se défendent, en prétextant que la demande de service contrevient aux contraintes de l'organisation. Et lorsque la demande de service est plus forte que les contraintes, les responsables de l'infrastructure se voient forcés de réagir dans les plus brefs délais.

Malheureusement, il n'y a pas de solution magique pour résoudre ce problème. Pour réduire l'insatisfaction, il faut se tenir informé des activités de sa clientèle, participer aux discussions stratégiques, prévoir les demandes potentielles, **exploiter le changement comme un avantage compétitif pour le client** et **satisfaire le client en livrant tôt et régulièrement des logiciels utiles**[1].

8.4.4 Processus

| **Valeur agile :** La réponse au changement davantage que le suivi d'un plan.

Les changements sont inévitables et imprévisibles. C'est particulièrement vrai pour les équipes dont la responsabilité est de résoudre les incidents survenant sur les systèmes en production.

1. Deux des principes agiles.

À partir de l'instant où les incidents sont reconnus comme inévitables, il ne reste plus qu'à trouver des moyens pour répondre de façon fiable et rapide à leur résolution. Le partage de connaissances, la documentation, les procédures d'intervention et des systèmes simples à maintenir sont autant de solutions pour augmenter la réactivité et le niveau de confiance des intervenants responsables des systèmes.

Un aspect plus complexe à aborder lorsqu'une équipe désire adopter une méthode itérative comme Scrum, c'est la planification des activités confrontée à leur capacité à s'engager sur un contenu déterminé à l'avance. Le contexte des équipes de support fait qu'il peut être frustrant de ne jamais être en mesure de tenir ses engagements en raison de l'apparition imprévisible des incidents.

La prochaine section décrit la méthode *Kanban*, qui offre une alternative aux équipes désirant adopter une méthode agile connue et cherchant à relever le défi de s'approprier une approche itérative.

8.5 LA MÉTHODE LEAN/KANBAN

> **Principe agile :** Les processus agiles promeuvent un rythme de développement durable.

La planification itérative peut être difficile, voire impossible pour certaines équipes, notamment parce que :

- Des spécialistes composent l'équipe et ils ne sont pas en mesure de partager leurs tâches avec leurs collègues, et vice versa. Par exemple, un graphiste n'est pas nécessairement en mesure de programmer et un développeur n'a pas toujours les compétences pour utiliser les outils de conception graphique. Cela limite la possibilité de créer des équipes multidisciplinaires capables de réaliser collectivement leurs engagements.
- Les responsabilités de soutien d'une équipe impliquent de répondre à des demandes survenant à l'improviste et compromettant le respect de leurs engagements.

L'échec régulier aux engagements n'a rien de motivant et les équipes ne pouvant pas s'adapter à la planification itérative sont déçues de la difficulté d'adopter une méthode comme Scrum. C'est particulièrement vrai pour les équipes d'infrastructure. C'est parfois le cas pour des équipes de développement, responsables de faire le soutien de la version en production tandis qu'elles travaillent à développer la version suivante, et que la demande du soutien est telle qu'elle provoque le non-respect des engagements.

La présente section décrit sommairement l'utilisation du tableau *Kanban*, un outil de la méthode *Lean*. Plusieurs des concepts décrits sont tirés de l'ouvrage *Kanban and Scrum – Making the most of two* des auteurs Henrik Kniberg et Mattias Skarin[1].

1. Henrik Kniberg et Mattias Skarin, *Kanban and Scrum – Making the most of two*, http://www.infoq.com/minibooks/kanban-scrum-minibook, consulté en décembre 2009.

8.5.1 La méthode agile la moins restrictive

La méthode *Lean* est une méthode agile de développement logiciel inspirée des méthodes manufacturières du « juste-à-temps ». En plus de principes du manifeste agile, cette méthode repose sur les principes suivants :

- l'élimination du gaspillage (voir le pour les sept formes répertoriées de gaspillage) ;
- l'apprentissage et l'amélioration continue ;
- la prise de décision au dernier moment responsable ;
- la livraison la plus rapide possible ;
- des équipes imputables et responsables ;
- des solutions intègres (simples, cohérentes, utiles et évolutives) ;
- la prise de recul et la nécessité d'un regard global sur la situation.

Tableau 8.1 — Les sept formes de gaspillage transposées du contexte manufacturier au contexte du développement logiciel[a]

Gaspillage dans un contexte manufacturier	Gaspillage dans un contexte de développement logiciel
1. Inventaire	1. Travaux incomplets
2. Surproduction	2. Caractéristiques ou fonctionnalités superflues
3. Traitement superflu	3. Réapprentissage (*relearning*)
4. Mouvements	4. Permutation entre les tâches
5. Transport	5. Transferts (de tâches, de composantes)
6. Attente	6. Délais
7. Défauts	7. Anomalies et corrections

a. Tiré et traduit de *Lean Software Development: An Agile Toolkit,* par Mary et Tom Poppendieck, Addison Wesley, 2003.

La méthode *Lean*, jumelée à l'outil du *Kanban*, est la méthode agile la moins prescriptive de toutes, car elle :

- n'impose aucun rôle, comme le responsable de produit ou le Scrum Master, ni de rencontres, comme la mêlée quotidienne ou la rétrospective ;
- n'a pas l'objectif de former des équipes multidisciplinaires et s'adapte à la participation des spécialistes ;
- ne propose pas de stratégie de planification comme l'utilisation des itérations et des engagements protégés ;
- ne recommande pas de facteur de priorisation particulier, comme la valeur d'affaires.

Par contre, si une organisation voit de la valeur à tenir des mêlées quotidiennes, à affecter un responsable de produit et à planifier les demandes par ordre de valeur d'affaires, elle n'aura aucune difficulté à intégrer ces autres pratiques agiles pour enrichir et adapter la méthode *Lean*.

8.5.2 Le tableau *Kanban*

Le mot *Kanban* désigne le tableau conçu par les entreprises manufacturières dans le but de réduire le gaspillage par l'utilisation d'un processus « juste-à-temps » ou à flux tirés[1]. Ces mêmes techniques sont aujourd'hui appliquées dans les processus de développement logiciel, avec les mêmes objectifs de limiter l'effort, réduire le gaspillage et contrôler le travail en cours (TEC[2]).

Le tableau sert d'outil permettant de surveiller le débit de production, d'identifier les goulets d'étranglement et de mesurer la productivité du processus. La Figure 8.1 présente l'exemple classique d'un tableau *Kanban*.

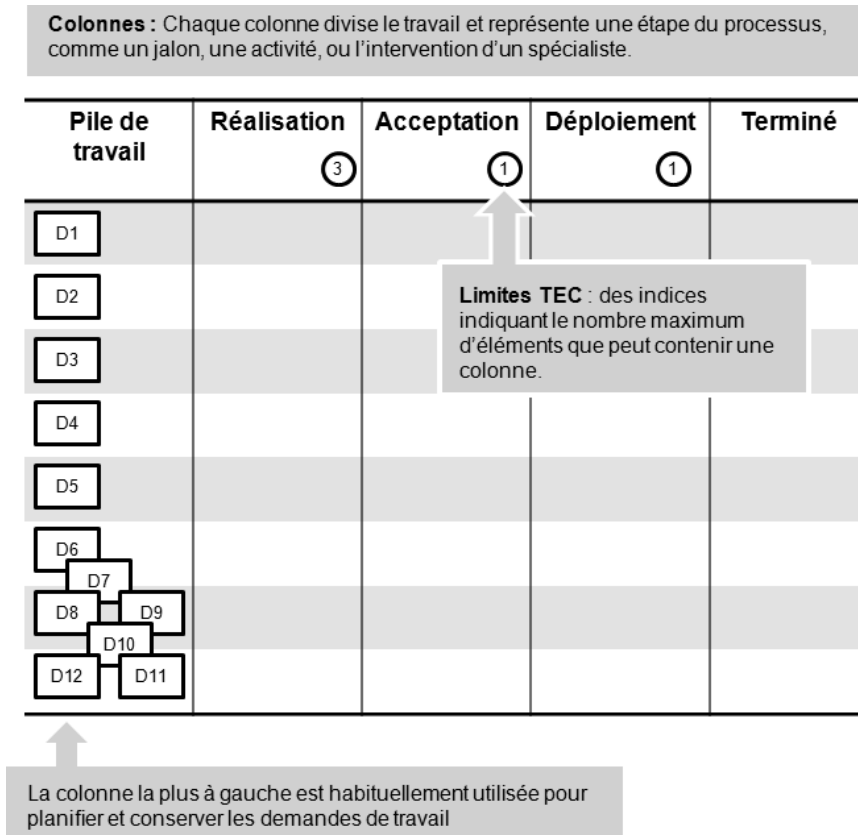


Figure 8.1 — Exemple d'un tableau *Kanban*.

1. Flux tiré : Modèle de pilotage des flux selon lequel les matières premières ou les pièces n'avancent dans la chaîne de fabrication que lorsque la demande les réclame. C'est donc une demande réelle qui va déclencher une production qui sera tirée en aval. Extrait de http://www.granddictionnaire.com, consulté en janvier 2011.
2. Aussi appelé *Work In Progress* (WIP).

À l'exemple du tableau des tâches de la méthode Scrum, chacune des colonnes du tableau représente un état des demandes et/ou des activités du processus. Les demandes seront déplacées d'une colonne à l'autre selon leur état d'avancement. Les colonnes de gauche et de droite servent respectivement à conserver les demandes travail en attente et accumuler les demandes terminées.

Il est possible d'affecter une colonne à un spécialiste précis pour observer le débit de travail autour de ses activités. D'ailleurs, il n'y a pas de règles particulières sur le nombre ou le type de colonnes composant le tableau. Nous conseillons d'ajouter autant de colonnes nécessaires pour isoler et mettre en évidence les inefficacités soupçonnées du processus.

Une particularité du *Kanban*, c'est la limite optionnelle restreignant le nombre d'items simultanés pouvant être contenus dans une même colonne. Par exemple, la colonne « Réalisation » de la indique qu'il n'est pas possible de réaliser plus de trois demandes en parallèle. Nous verrons comment cette limite permet de limiter le TEC.

8.5.3 La planification et la mesure de productivité

La méthode *Lean/Kanban* ne recommande pas de technique ou de vecteurs de planification en particulier. Son principe de fonctionnement est basé sur un flux de traitement continuel des items. Il est donc possible d'utiliser la valeur d'affaires, le niveau d'urgence, le temps d'attente, ou tout autre vecteur pertinent du domaine d'affaires.

Des règles doivent donc être établies avec l'équipe pour indiquer dans quel ordre les éléments de la pile doivent être entrepris. Quelqu'un, le client par exemple, peut être responsable de tenir la pile ordonnée. Dans ce cas, la règle pour choisir la prochaine tâche est de toujours prendre le premier élément de la pile. Les items peuvent être réorganisés à tout moment, ce qui permet de planifier en haute priorité n'importe quelle demande imprévue.

Pour réduire et éliminer le gaspillage en cours de processus, les équipes mesurent le temps moyen de cycle avec un tableau *Kanban*. Il représente le temps observé entre le moment où les items sortent de la pile de travail et celui où ils atteignent la dernière colonne.

Une équipe voulant améliorer sa productivité cherchera à réduire son temps de cycle, en diminuant le temps de transition entre les colonnes et les délais d'intervention à chaque colonne.

L'estimation des items est également optionnelle. Cependant, si les items ne sont pas de taille comparable, il est utile de les estimer afin de calculer un temps de cycle prenant en compte la complexité des items. Dans ce cas, il est pertinent de faire une estimation par affinité avec une échelle de points d'effort. Par contre, il est plus simple d'améliorer le débit de production en découpant les items de façon à obtenir des tailles similaires. Une équipe désirant améliorer sa prévisibilité cherchera à réduire la variance du temps de cycle.

8.5.4 Le travail en cours et les goulets d'étranglement

Limiter le TEC permet de contrôler le gaspillage de l'effort causé par le travail en parallèle et ainsi réduire le temps de cycle des items. C'est avec les limites d'items par colonne qu'il est possible d'identifier les inefficacités bloquant ou freinant la chaîne de traitement. En effet, lorsque des items restent bloqués dans une colonne, un blocage ou une difficulté est mise en évidence.

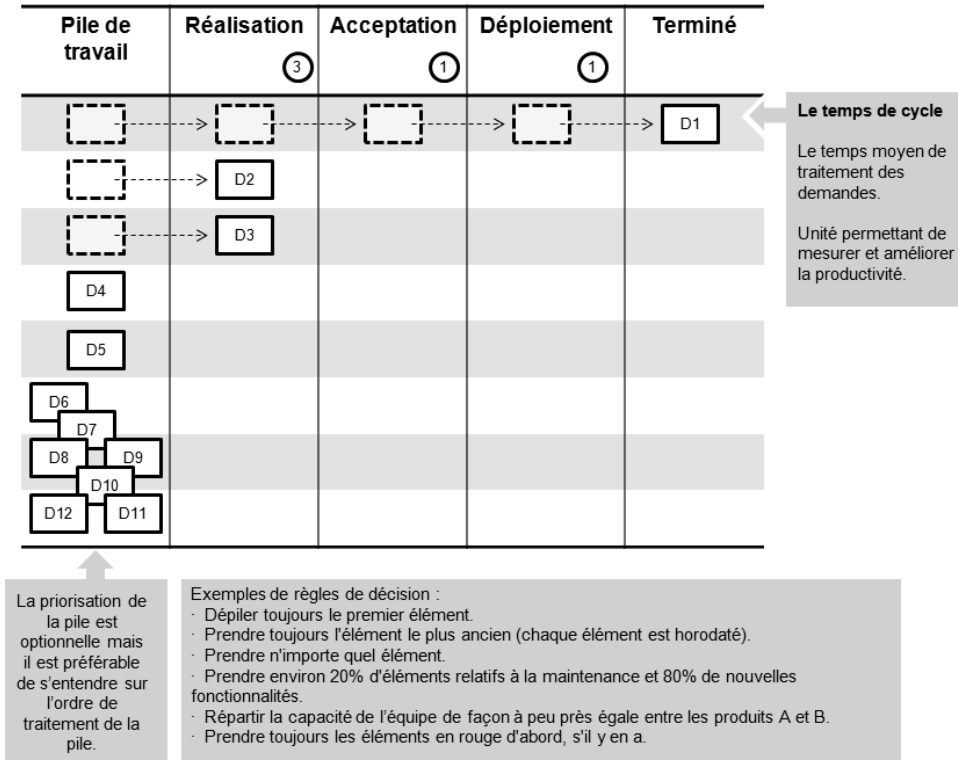


Figure 8.2 — Description de la planification et la mesure de productivité.

Le principe est qu'il est inutile de travailler sur de nouveaux items si le processus doit être « réparé ». Plutôt que d'ignorer les problèmes, la technique du *Kanban* propose d'arrêter le travail en cours le temps de résoudre les incidents à la source.

Le blocage est une pratique prévue par le tableau *Kanban*. Elle peut sembler contraignante au départ, mais sur le long terme, elle prévient la réapparition des problèmes et diminue la variance du temps de cycle ainsi que le TEC. Limiter le TEC est un des objectifs des principes du « juste-à-temps » où on considère tous travaux commencés mais non terminés comme un gaspillage potentiel, ne répondant pas aux conditions de succès. Pour réduire cet inventaire de gaspillage potentiel, il est conseillé de limiter au maximum le TEC et de livrer le plus rapidement possible.

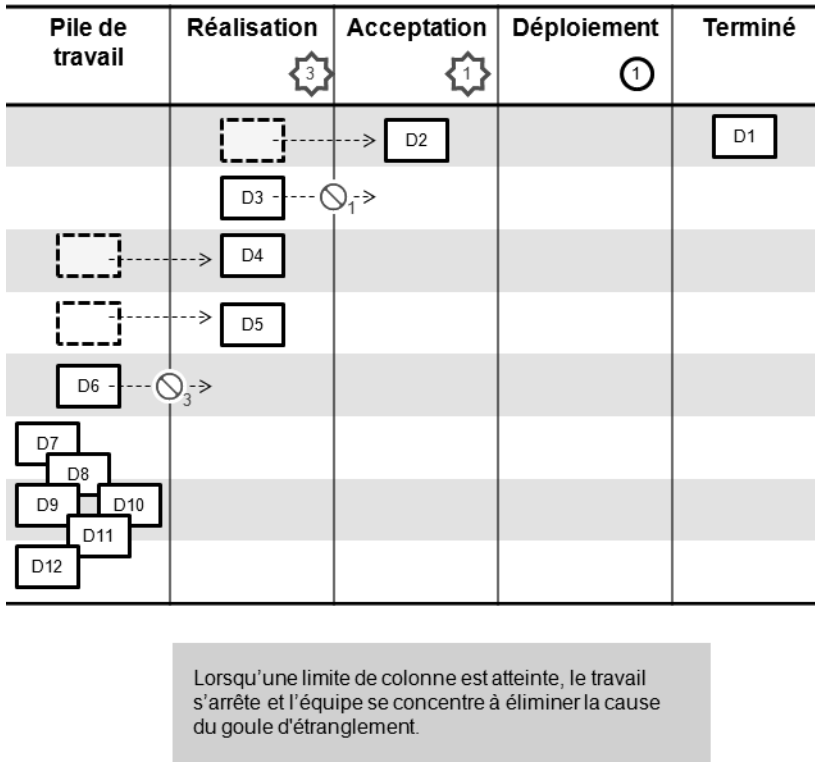


Figure 8.3 — Les limites des colonnes permettent de mettre en évidence les inefficacités du processus.

Certaines équipes n'utilisent pas la règle du blocage, la trouvant trop contraignante. Ce qui ne les empêche pas de considérer que tout dépassement de limite est une piste d'amélioration devant être explorée dès que possible.

Conseil : Les limites devraient être fixées de manière à proposer un rythme soutenu et soutenable aux membres des équipes. L'idée est d'obtenir à terme un débit de production fluide, et non pas de mettre une pression excessive.

Le travail sous pression est la cause fréquente de la mauvaise qualité et de l'introduction de régression. Ceci est particulièrement vrai pour les équipes d'infrastructure ayant souvent des contraintes d'horaires spéciales et des délais de travail serrés pour limiter les impacts de l'indisponibilité des services à la clientèle.

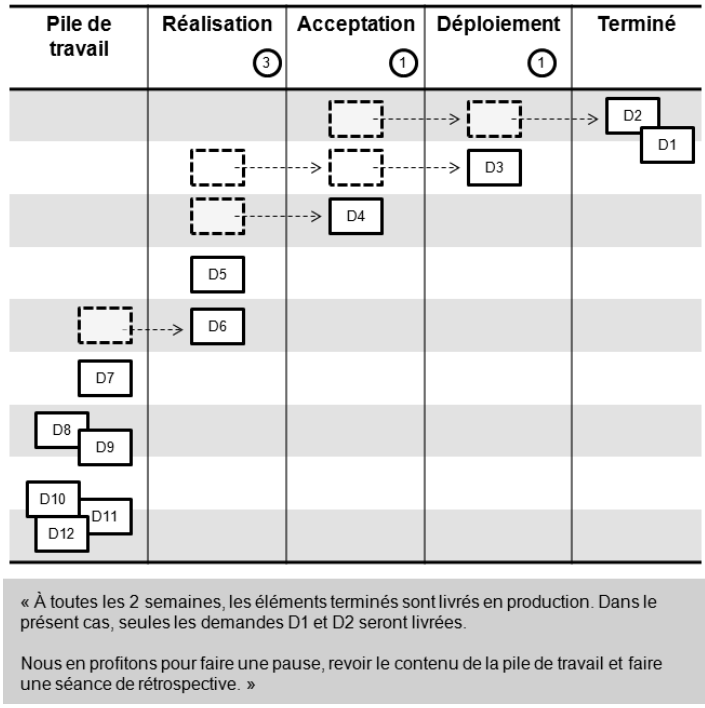


Figure 8.4 — Exemple des règles de livraison d'une équipe d'infrastructure.

8.5.5 La livraison

La méthode *Lean/Kanban* ne propose pas de règles spécifiques pour livrer les items terminés en production. Il est donc possible de définir la règle répondant le mieux aux besoins de la clientèle comme suit :

- À période fixe (par exemple, toutes les quatre semaines), les éléments terminés sont installés. Cela a l'avantage de rendre prévisibles les moments de livraison.
- Tous les *x* éléments : pour éviter l'accumulation d'un trop grand nombre d'élément à livrer et ainsi restreindre le champ d'investigation si un problème survient.
- À chaque élément : pour faire du « juste-à-temps », à condition que les mécanismes de livraison possèdent l'efficacité requise.
- À la demande : pour accommoder la transition au changement.
- Ou toute autre règle convenant au contexte de l'organisation.

Pour terminer, si les principes *Lean* vous intéressent et que vous désirez approfondir votre compréhension du tableau *Kanban*, nous vous invitons à vous référer à l'ouvrage d'Henrik Kniberg et Mattias Skarin, ainsi qu'aux différentes publications de Mary et Tom Poppendieck, David Anderson et Mike Sullivan, des spécialistes de la méthode *Lean* et du tableau *Kanban*.

En résumé

La collaboration entre les équipes de développement et les responsables de l'infrastructure est une relation fragile et l'adoption de l'agilité ajoute de la pression sur cette relation. Même si les équipes de développement peuvent isoler momentanément les impacts de l'agilité sur leurs confrères, tôt ou tard le mode de collaboration devra être ajusté.

Les équipes d'infrastructure doivent s'adapter aux impacts apportés par l'agilité, comme la préparation rapide de multiples environnements, l'invitation des équipes de développement à participer aux travaux d'infrastructure, la livraison fréquente des solutions logicielles, l'adoption d'une définition de terminé et l'utilisation des conditions de succès.

Les équipes de développement doivent apprendre à considérer les administrateurs de système comme des parties prenantes des produits qui ont des attentes et des besoins en matière de maintenabilité.

Les équipes d'infrastructure désirant faire le virage agile à leur tour devront concentrer leur transition autour de certains principes agiles, comme de préconiser la simplicité, la communication, la collaboration et revoir leur processus.

En particulier, le processus de la gestion des demandes, auquel Scrum ne convient pas lorsque :

– les spécialistes ne peuvent pas partager leurs travaux ;

– les demandes *ad hoc* ou urgentes brisent constamment les engagements.

Il est alors possible de choisir une approche agile plus adaptée à la réalité des équipes d'infrastructure, notamment le *Lean/Kanban*.

9

Entretien et évolution des applications

Objectif

Dans ce chapitre, vous découvrirez l'essentiel des pratiques agiles liées à l'entretien et l'évolution d'un système déployé en production. Vous serez en mesure d'identifier les différents types de requêtes de modification. Vous serez capable de choisir le mécanisme de gestion de demandes de changement le plus adapté au contexte de vos projets. Finalement, vous apprendrez à utiliser les défauts comme source d'amélioration du processus de développement.

Mise en situation : Éric et la maintenance d'un portfolio d'applications

Éric est directeur d'un département des TI d'une société de vente au détail de produits pharmaceutiques et d'hygiène personnelle. Leurs clients sont les 2 000 franchisés à travers le pays. Leur parc applicatif comporte des systèmes développés sur quatre plateformes différentes, dont certains doivent échanger de l'information. La plupart des systèmes sont découpés en plusieurs modules qui comblent chacun un ensemble de besoins liés aux processus d'affaires. Le département des TI compte environ 150 personnes.

Éric reçoit les requêtes de modification en provenance du comité des TI et les priorise en fonction de leur valeur d'affaires et de leur capacité à permettre d'atteindre la vision corporative. Une partie du travail d'Éric consiste à faire analyser brièvement chaque requête pour comprendre l'impact qu'elle aura sur les systèmes existants et

avoir une idée, même très vague, du coût de réalisation. Ces requêtes de modification contiennent :

– des créations de nouvelles applications suite à l'émergence d'un besoin d'affaires ;

– des ajouts, modifications et suppressions de fonctionnalités de systèmes existants ;

– des corrections de défauts de toutes sévérités (critiques, importantes et mineures).

Les requêtes dont l'effort de réalisation est inférieur à 20 jours-personnes sont placées dans la liste des requêtes approuvées de l'équipe responsable de l'application concernée. Cette liste est déjà priorisée et chaque équipe d'entretien réalise la requête de modification suivante quand elle est de nouveau disponible. Éric fait un suivi une fois par mois avec chacune des équipes afin de rendre compte de l'avancement des travaux au comité des TI. Ces « petites » requêtes de modification représentent plus de 70 % de l'effort de réalisation dépensé annuellement par ses équipes.

Les requêtes dont l'effort de réalisation dépasse 20 jours-personnes sont ciblées en tant que projet, avec une équipe constituée d'un chargé de projet, d'un analyste principal et des membres d'équipe. Dans ces cas, le « processus de développement des projets » de l'organisation s'applique intégralement. Ce processus est une méthodologie développée par eux au fil du temps. Il arrive aussi que plusieurs requêtes de modification impactant le même module d'un système soient réunies pour constituer un projet de développement.

Éric songe à adopter l'agilité dans ses projets de développement afin de livrer des applications qui apporteraient encore plus de valeur aux clients. Son personnel est intéressé et motivé par cette nouveauté mais on comprend que 70 % du travail se fait sous la forme traditionnelle qu'on lui connaît.

Éric constate qu'il possède déjà un des ingrédients essentiels : un carnet de produit priorisé et estimé. La transition à l'agilité s'annonce plutôt simple et déjà Éric entrevoit une hausse de la motivation du personnel affecté à l'entretien et l'évolution des systèmes.

« Dans la vie, vous aurez trois certitudes :
1- Vous payerez des taxes ;
2- Nous mourrons tous un jour ;
3- Il y aura des changements.
Et ce dernier point est particulièrement applicable en développement logiciel. »
- Nick Papiccio, manager de génie logiciel[1]

1. Nick a été mon supérieur pendant plus de trois ans dans la deuxième moitié des années 1990. Il a été un mentor pour nous tous dans son équipe, dont le nombre est passé de 23 à 110 personnes durant cette période. Ses enseignements me servent toujours aujourd'hui.
– Sylvie Trudel.

9.1 LES INCIDENTS, LES PROBLÈMES ET LES REQUÊTES DE MODIFICATION

Du point de vue du développement et de l'entretien des applications, les organisations travaillent souvent en deux modes : projet ou service. Les précédents chapitres ont expliqué comment s'applique l'agilité en mode projet. Pendant le projet, les pratiques visent à limiter les incidents qui surviendront suite à la livraison des applications. Cependant, même en portant une attention particulière à la qualité technique et fonctionnelle des livrables, ces incidents ne seront jamais totalement éliminés.

En mode service, aussi appelé « mode opérationnel », les applications ont été livrées aux utilisateurs et intégrées à l'infrastructure de l'organisation. Toutes sortes d'incidents peuvent alors survenir au cours de leur utilisation.

Quelques définitions[1] ...

– **Incident** : un incident est une difficulté imprévue survenant au cours de l'utilisation d'un système. Un incident peut être de nature variée, allant d'une simple question d'un utilisateur sur le fonctionnement d'une nouvelle imprimante, jusqu'à une panne stoppant les opérations de l'organisation.

– **Défaillance** (synonyme : panne) : une défaillance est l'exécution (ou la manifestation) d'une faute dans l'environnement opérationnel. Une défaillance se définit comme la fin de l'aptitude d'un composant à exécuter une fonction dont il est totalement ou partiellement responsable. L'origine d'une défaillance repose dans un défaut qui est tapi dans le système opérationnel. Tant que le système en production n'exécute pas l'instruction (ou traite une donnée) fautive, il fonctionne normalement. Il est donc plausible que vos systèmes opérationnels comportent des défauts qui n'ont pas encore été exécutés.

– **Défaut** (synonyme : faute) : les défauts sont issus d'erreurs humaines qui n'ont pas été détectées lors du développement ou de la modification du logiciel. Une erreur peut être grammaticale, de logique ou de données.

Pour assurer le soutien aux affaires de l'organisation, il est possible que les processus de gestion de l'infrastructure soient inspirés du modèle ITIL © (*Information Technology Infrastructure Library*), qui contient des bonnes pratiques de gestion de l'infrastructure, notamment celles de la gestion des incidents. Ces pratiques sont souvent mises en œuvre par un groupe de support aux utilisateurs – appelé aussi service à la clientèle ou bureau d'aide –, que ce dernier soit interne ou externe à l'organisation. La performance d'un tel groupe est souvent mesurée par la « qualité de service » offert, dont l'un des indicateurs est parfois défini par le temps moyen pour résoudre un incident, en fonction de sa sévérité.

Pour assurer la qualité de service attendue, ce groupe doit faire un suivi efficace des incidents qui lui sont rapportés par les utilisateurs. Lorsqu'un incident est rapporté, il est analysé pour déterminer sa sévérité et sa priorité en fonction de l'impact sur

1. Traduit et adapté de la norme ISO/IEC 14764 sur la maintenance du logiciel.

les opérations ou les affaires. Il est fréquent de rencontrer des mécanismes d'escalade permettant de mieux résoudre ces incidents. Ainsi, quand un volume important d'incidents survient, le groupe de support est plus facilement en mesure de traiter les incidents critiques en premier, en plus d'aiguiller la résolution vers les personnes ayant les compétences requises.

Un incident peut avoir plusieurs états, tels qu'*identifié*, *assigné*, *en traitement*, *en attente* et *résolu*. Puis, selon la nature de l'incident ou si le service à la clientèle n'arrive pas à le résoudre rapidement, un incident peut être transformé en rapport de problème ou en requête de modification. Les incidents rapportés concernent soit une composante de l'infrastructure, soit l'une des applications utilisées, notamment celles qui sont développées ou entretenues par l'organisation et dont il est question dans ce chapitre.

Lorsqu'un rapport de problème est identifié, il est analysé et un état lui est attribué en fonction de la phase de son cycle de résolution, similaire aux états définis pour les incidents. Dans plusieurs cas, les rapports de problèmes peuvent être résolus sans modification au logiciel. Par exemple, en changeant une donnée de configuration ou un paramètre, ou en redémarrant une composante logicielle en panne. Dans d'autres cas, la résolution du problème demande de modifier le logiciel et une requête de modification ou un rapport de problème est alors créé (voir la Figure 9.1 pour les transformations possibles d'un incident). Une requête de modification peut aussi provenir d'un utilisateur désirant qu'une amélioration soit apportée au logiciel.

Que ce soit un rapport de problème ou une requête de modification, une priorité doit leur être attribuée. Un rapport de problème est urgent parce qu'il y a une défaillance opérationnelle sans possibilité de la contourner, il est normal de laisser tomber les travaux en cours (sur les requêtes de modifications) pour s'attaquer à ce problème afin de le résoudre sans délai.

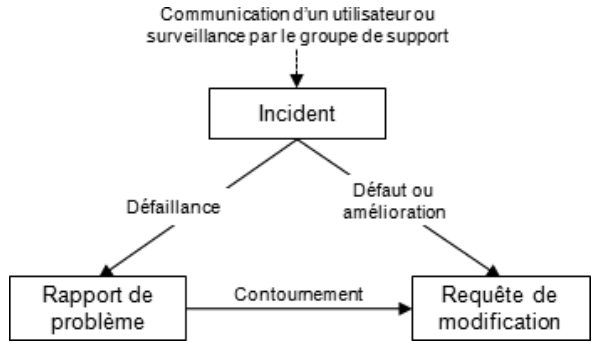


Figure 9.1 — Transformations possibles d'un incident.

Une organisation appliquant un processus en cascades a l'habitude de transférer les rapports de problèmes et les requêtes de modification à une équipe de maintenance, qui en assure la gestion et le traitement, parce que l'équipe de développement a été démantelée et ses membres réaffectés à d'autres projets.

En adoptant l'agilité, il est fort probable que l'application soit mise en production bien avant que l'ensemble du projet soit terminé. Cette application partielle est néanmoins utilisée pendant que l'équipe continue de développer des fonctionnalités, tout en étant engagée à livrer un contenu défini à chaque itération. L'effort requis pour gérer et traiter les problèmes et les requêtes de modification peut mettre en péril la capacité de l'équipe à tenir cet engagement.

Selon le volume et l'urgence des rapports de problèmes et des requêtes de modification, divers mécanismes de gestion peuvent être adoptés par les équipes ; ils sont discutés à la section 9.3. Mais au préalable, il est utile de bien classifier les rapports de problèmes et requêtes de modification pour alimenter ces mécanismes de gestion. C'est ce dont nous traitons ci-après.

9.2 LES TYPES DE MAINTENANCE DU LOGICIEL

L'entretien et l'évolution des applications sont présentés dans la littérature comme des activités de « maintenance du logiciel ». Selon la norme ISO/IEC 14764, la maintenance se découpe en quatre types distincts, tel qu'illustré à la Figure 9.2.

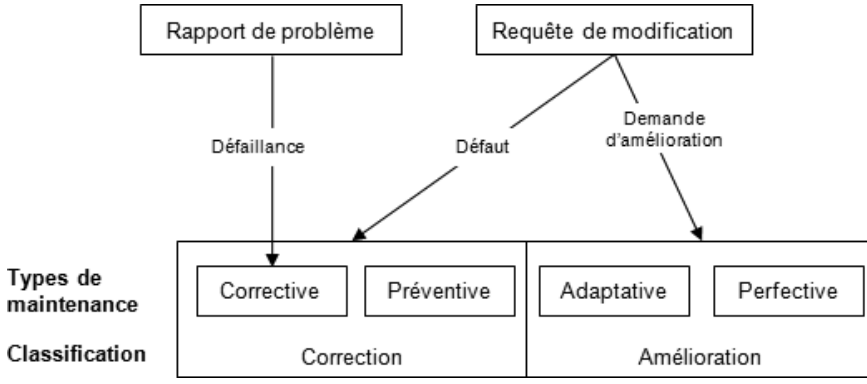


Figure 9.2 — Les types de maintenance.

- **Maintenance perfective** : les besoins d'affaires ont évolué ou une opportunité d'amélioration a été identifiée, comme un gain d'efficacité, une augmentation de la productivité, ou une optimisation des processus opérationnels. Des fonctionnalités doivent alors évoluer pour combler ces nouveaux besoins ou opportunités. Aussi, l'équipe de développement, avec l'appui du responsable de produit, décide de reconstruire une portion de l'application (restructuration de code) afin d'améliorer sa maintenabilité (ou tout autre critère de qualité interne) qui aura pour effet d'accélérer la vélocité de l'équipe ou de préparer l'application pour un changement évolutif imminent.
- **Maintenance adaptative** : une application maintenue par l'organisation doit être modifiée pour demeurer utilisable car une composante de l'infrastructure

avec laquelle elle s'interface est ou sera changée. Par exemple, le choix d'un système de gestion de la base de données comme nouvel outil d'infrastructure peut obliger l'adaptation de la couche d'accès aux données de l'application pour demeurer à jour.

- **Maintenance préventive** : l'équipe de développement doit faire la correction d'un défaut latent qui, s'il n'est pas corrigé, fera un incident en production bientôt. Par exemple, le célèbre problème du passage à l'an 2000, où l'ensemble des systèmes manipulant des dates ont dû être soumis à de la maintenance préventive car les variables contenant l'année n'avaient souvent que deux chiffres au lieu de quatre.
- **Maintenance corrective** : des défauts entravent les opérations et les affaires requérant des modifications dans l'application.

Les activités de maintenance peuvent être de différentes tailles selon la nature du besoin, allant d'un simple changement de libellé d'une fonctionnalité à l'écran jusqu'à la création d'un système complet. Il est alors normal de vouloir catégoriser ou classifier chacune des requêtes de modification afin d'alimenter leur mécanisme de gestion.

Ensuite, ce n'est pas parce qu'un utilisateur fait une requête de modification qu'elle sera systématiquement mise en œuvre. Il doit y avoir un mécanisme pour accepter ou refuser une requête. April et Abran, dans leur ouvrage *Améliorer la maintenance du logiciel*, proposent le processus d'acceptation ou de refus du travail de la maintenance.

Tous les rapports de problèmes sont traités *illico*, aucun refus possible puisque les opérations sont en panne ou fortement incommodées. Quand il existe un contournement pour une défaillance, sa correction peut être considérée moins urgente, auquel cas l'incident sera transformé en requête de modification. Toutefois, la recherche de la solution de contournement doit se faire rapidement pour déterminer l'urgence de la correction. Pour toutes les autres requêtes de modification considérées non urgentes, un type de maintenance peut être assigné et la requête de modification est insérée dans la liste des requêtes. En mode agile, c'est le responsable du produit qui continue de prioriser chacun de ces items et les requêtes sont insérées dans le carnet de produit. Ce même responsable de produit peut également refuser une requête et communiquer ses raisons au requérant.

Quand plusieurs requêtes concernent un groupe de fonctionnalités, il peut être souhaitable de les regrouper afin de les traiter dans une même livraison de l'application. Ce regroupement peut représenter plus de « n jours » de travail, soit l'équivalent du nombre de jours pour constituer un projet. Chaque organisation doit définir la valeur de ce « n jours ». Toutefois, nous recommandons que cette valeur soit plus élevée que le coût de lancement du projet. Selon notre expérience, cette valeur se situe entre cinq et cinquante jours, selon la taille de l'organisation et l'efficacité de son processus de démarrage de projet.

9.2.1 Les fausses demandes d'amélioration

La catégorisation des défauts suscite souvent de la confusion et des discussions animées. Lorsqu'on évolue dans les cultures de contrôle ou de compétence, il est fréquent que

des défauts soient classifiés comme des demandes d'amélioration, de façon à ce que les indicateurs sur le nombre de défauts par livraison paraissent bien. Cette catégorisation se base sur une définition des défauts ressemblant à : « *tout comportement de l'application non conforme à ce qui est écrit dans les documents d'analyse* ». Cette définition fait en sorte que toute omission dans les documents d'exigences est interprétée comme des besoins non spécifiés, donc une demande d'amélioration. Alors imaginez la multiplication de requêtes de modification évolutive quand les équipes documentaient les exigences de façon exhaustive avant l'agilité, et qu'ils choisissaient de diminuer les écrits en adoptant les scénarios utilisateurs avec l'agilité.

Conseil : Avec l'adoption de l'agilité, il serait bon de revoir la définition d'un défaut et d'une demande d'amélioration, surtout si la façon de documenter les exigences a aussi changé.

D'autres équipes multiplient plutôt le nombre de défauts en identifiant toute spécification non complétée au terme d'une itération comme une requête de modification. Cette catégorisation a pour but de légitimer le travail non complété pendant l'itération, qui doit maintenant être planifié dans une itération ultérieure. Dans ce cas, l'équipe manque de transparence sur le travail réellement accompli puisqu'il reste des défauts à corriger pour considérer l'item du carnet de produit comme étant « terminé ». Ainsi, les apparences sont sauvées car le calcul de leur vélocité a été gonflé.

Les indicateurs de mesure, et surtout leur interprétation perçue dans une culture donnée, peuvent pousser les équipes à manquer de transparence et à adopter un mauvais système de catégorisation afin de paraître sous leur meilleur angle. L'équipe doit comprendre, qu'indépendamment de la catégorisation, il faut éviter de surcharger le carnet de produit de corrections ou de demandes d'amélioration qui, au final, ne mesurent pas le véritable état d'avancement des travaux de maintenance et de développement.

Comme le client est responsable de la planification des travaux de développement, ce dernier doit être vigilant à cette catégorisation et l'équipe doit reconnaître son droit à refuser toute livraison de fonctionnalités n'étant pas conforme à la spécification entendue au moment de la planification. Au moment de la mise en production, ce n'est pas le nombre de défauts corrigés ou de requêtes d'amélioration mises en œuvre qui compte, mais la livraison d'un système utile, fonctionnel et cohérent.

9.3 LES MÉCANISMES DE GESTION DES REQUÊTES ET DES PROBLÈMES EN MODE AGILE

Dans un contexte agile, nous avons observé différents mécanismes de gestion des requêtes de modification et des rapports de problèmes en fonction de leur nombre, de l'effort requis pour les traiter dans des délais raisonnables ou du contexte des équipes et des projets :

- la gestion par carnet de produit ;
- le « *SWAT team* » ;
- la création d'un projet de maintenance ;
- le Kanban.

Ces mécanismes ne sont pas mutuellement exclusifs puisqu'ils peuvent être combinés. Cependant, ils traitent de considérations différentes et c'est pourquoi nous les avons distingués. Leurs contextes d'applicabilité sont décrits ci-après.

9.3.1 La gestion par carnet de produit

Le contexte

L'équipe de développement du produit logiciel est toujours active et elle a livré des versions de l'application en production.

Le nombre de requêtes de modification est petit et celui des rapports de problèmes est quasi nul. Il est alors possible de les gérer en mode projet, avec une planification itérative.

Comment ?

La gestion d'une petite quantité de requêtes de modification ne requiert pas d'outil de gestion particulier, ni de processus sophistiqué. Une requête de modification en cours de projet devient systématiquement un item du carnet de produit et ce dernier est traité comme n'importe quel autre item, dans le cadre du projet.

Les corrections se font au fur et à mesure, une à la fois. Tant qu'une fonctionnalité n'est pas conforme, elle n'est pas planifiée pour la prochaine livraison et le travail n'est pas considéré comme terminé.

Avec des livraisons fréquentes et prévisibles, il est possible de prévoir à quel moment une requête de modification sera traitée, ce qui permet de le communiquer au requérant.

9.3.2 Le « *SWAT team* »

Le contexte

Comme pour la gestion par carnet de produit, l'équipe de développement est toujours active. Elle est la seule responsable de la maintenance de la version en production, tout en continuant de développer de nouvelles versions. Mais contrairement à la gestion par carnet de produit, il n'est plus possible de prévoir ses engagements car l'effort requis pour corriger les rapports de problèmes est imprévisible. L'équipe veut se doter d'un moyen, d'une part pour reprendre des engagements prévisibles et d'autre part pour se consacrer sans contrainte de temps à l'entretien du système.

Comment ?

Dans ces cas, il est fréquent que les équipes de développement consacrent une portion de leurs heures disponibles pour corriger la version en production tandis qu'elles développent la prochaine version. Une des techniques à envisager est de dédier des plages horaires fixes (par exemple, tous les mercredis matins, ou tous les matins de 8 heures à 10 heures) comme période de correction.

D'autres équipes choisissent une technique où certains membres de l'équipe se portent volontaires à tour de rôle pour le traitement des rapports de problèmes et de la maintenance corrective. Chacun des membres du *SWAT team*[1] joue momentanément le rôle de mainteneur de l'application traitant en priorité tous les rapports de problèmes et requêtes de modifications urgentes qui surgissent durant l'itération.

Il faut cependant prendre garde à ce que ces règles ne nuisent pas à l'efficacité générale de l'équipe, développeurs et mainteneurs confondus. Par exemple :

- Lorsqu'un mainteneur ne parvient pas à résoudre un problème critique et urgent, il est normal que les développeurs s'arrêtent pour lui donner un coup de main.
- Lorsqu'un mainteneur a terminé les corrections, il est normal qu'il contribue à la réalisation de l'itération.

Le client est toujours responsable de la planification des corrections. Il doit donc être disponible pour évaluer qu'une correction en vaut la peine sans gaspiller les ressources disponibles au développement.

Si le volume de corrections ne se résorbe jamais, il est probable que le *SWAT team* soit maintenu perpétuellement. Mais il est également possible que ce mécanisme soit aboli suite à la révision des pratiques de développement augmentant la qualité des livraisons et limitant le volume des requêtes. L'abolition du *SWAT team* devrait d'ailleurs être un objectif du processus d'amélioration continu de l'équipe.

Considérations

Il arrive qu'un développeur se découvre une vocation de mainteneur et désire jouer ce rôle à plein-temps. Ces rares individus peuvent apporter une spécialité intéressante à l'équipe, car les développeurs préfèrent généralement la création de nouvelles fonctionnalités à la maintenance[2].

Malgré leur préférence, il est avantageux que tous les développeurs participent périodiquement à la maintenance de leur application. Les activités de maintenance ouvrent l'esprit des développeurs à la réalité des mainteneurs. Cette prise de conscience a de fortes chances de modifier leurs pratiques de développement de façon à prévenir la récurrence des problèmes et faciliter leur correction. Ce pourrait être, par exemple, par le développement de tests unitaires plus concluants ou par la production de lignes

1. Appelée aussi « équipe pompier » ou « équipe d'urgence ».

2. *« Devant l'afflux de volontaires, nous allons recourir aux chaises musicales pour désigner l'espion »* - Caius Bonus, dans *Astérix le Gaulois*, par Goscinny R. et Uderzo A., 6e édition, Éditions Hachette, 1999.

de journalisation plus explicites. Selon nous, la maintenance est une expérience essentielle pour devenir un excellent développeur.

9.3.3 La création d'un projet de maintenance

Le contexte

Quand les requêtes de modification surviennent, elles ne sont pas nécessairement traitées immédiatement, soit :

- pour des considérations de disponibilité des équipes ;
- par le peu de criticité de ces requêtes ;
- parce qu'il est difficile d'en établir la valeur d'affaires de façon isolée.

Alors, un inventaire de requêtes s'accumule. La mise en œuvre de chacune est difficilement justifiable, mais une valeur d'affaires peut se dégager en les regroupant en sous-ensembles cohérents.

Mise en situation : Pierre et les requêtes d'amélioration

La PME de Pierre développe des applications en sous-traitance pour une grande société financière. Plus de 350 utilisateurs se servent de leurs 30 modules à travers le pays et ils peuvent soumettre des requêtes d'amélioration conservées dans un carnet.

Quand les requêtes ont trait à un nouveau besoin d'affaires et qu'elles sont d'envergure considérable, elles sont extraites du carnet des requêtes pour être placées dans le carnet des projets. Les requêtes de modification qui restent dans le carnet des requêtes sont donc toutes de petite envergure et non critiques aux affaires.

Avec une disponibilité grandissante de certaines de ses équipes de développement, Pierre a enfin eu l'opportunité de s'occuper des requêtes accumulées. Au fil des années, le carnet des requêtes a grossi jusqu'à contenir 1 200 items (nouvelles fonctionnalités, ajout d'options aux fonctionnalités existantes, création de raccourcis ou d'aide à la saisie, etc.). Avant de commencer, le carnet doit être revu afin d'exclure les requêtes qui ne sont plus pertinentes et identifier celles ayant le plus grand retour sur investissement, pour ensuite les regrouper dans des projets de maintenance.

Pour mesurer le retour sur investissement, il a été décidé d'estimer le gain d'efficacité opérationnelle apporté par chacune des requêtes et de le comparer à l'effort de réalisation requis. Le ratio entre le gain et le coût a servi de coefficient pour extraire les requêtes à mettre en œuvre. Pour chacune des requêtes, les données suivantes ont été utilisées :

– fréquence d'utilisation : en se fiant au volume annuel de transactions – générées dans la base de données par la fonctionnalité touchée ;

– estimation de l'effort (en minutes) économisé par transaction par les utilisateurs, si la requête est mise en œuvre ;

– gain d'efficacité opérationnelle (en heures et en argent) : fréquence x effort estimé ;

– coût estimé de réalisation (en heures et en argent) fourni par l'équipe de Pierre ;

– ratio du retour sur investissement : gain d'efficacité opérationnelle/coût estimé de réalisation.

Cette technique a permis de constituer un projet par module avec les requêtes à valeur ajoutée, excluant celles dont le coût de réalisation était plus élevé que le gain anticipé (ratio inférieur à 1).

Les résultats de cet exercice ont introduit la sélection automatique des requêtes d'amélioration, en calculant dès leur soumission le retour sur investissement. Cela a grandement amélioré la gestion du carnet des requêtes en limitant l'inventaire.

Comment ?

La mise en situation présentée dans l'encadré précédent souligne qu'il est parfois souhaitable de créer un projet pour traiter un inventaire important de requêtes. Dans ce cas, les critères et pratiques discutés au chapitre 3 – *Démarrage d'un projet agile* sont valides :

- rédaction d'une charte de projet dont les objectifs du projet illustrent la valeur ajoutée ciblée ;
- constitution d'une équipe de projet ;
- élaboration d'un carnet de produit à partir des requêtes de modification ;
- préparation d'un plan de livraison.

9.3.4 Le *Kanban*

Le contexte

Lorsque la priorité changeante des requêtes de modification et des rapports de problèmes empêche une équipe de travailler en mode itératif, l'équipe de maintenance doit modifier sa méthode de gestion agile. Par exemple, elle peut adopter la technique du *Kanban* où la planification itérative est optionnelle. Cette technique convient aux équipes incapables de s'engager à livrer un contenu déterminé dans un délai fixé.

Comment ?

Cette technique peut se combiner à chacun des trois mécanismes de gestion cités précédemment. Nous vous invitons à vous référer au chapitre 8 – *Infrastructure technologique* pour une description plus détaillée du *Kanban* et évaluer si ce mode de gestion est mieux adapté pour vos besoins et votre contexte.

9.4 LA PRÉVENTION DES DÉFAUTS

L'identification précise de la cause d'une défaillance est la principale difficulté afin d'estimer l'effort de correction requis. Ce problème important est particulièrement apparent dans les périodes de stabilisation qui précèdent les mises en production. Il n'est pas rare de constater des périodes de stabilisation qui s'étirent au point de provoquer le dépassement des budgets et des échéances d'un projet. Un cercle vicieux s'installe alors : plus de défaillances sont identifiées, et plus il y a nécessité de modifier le logiciel et d'exécuter à nouveau les plans de tests. Généralement, les plans de tests

améliorés détectent encore plus de défauts, ce qui réduit de plus en plus le temps disponible pour modifier et tester à nouveau le système qui devrait normalement être prêt pour la production.

L'effort de correction a tendance à ralentir les équipes de développement : notre expérience démontre que les équipes devant appliquer un certain nombre de corrections ont une vélocité[1] mesurée plus basse que les équipes qui n'ont presque pas de corrections à appliquer. Une correction requiert un effort parfois considérable pour être traitée de bout en bout et cet effort n'apporte pas de valeur d'affaires puisque la fonctionnalité était censée fonctionner tel que requis dès le départ. Par contre, ne pas traiter la correction enlève de la valeur à l'application puisque la valeur d'affaires associée à la fonctionnalité inadéquate ne peut pas être acquise par les clients. Autrement dit, contrairement à une amélioration fonctionnelle, la taille relative d'une correction ne devrait pas être prise en compte dans le calcul de la vélocité, car sa seule valeur est de rectifier un logiciel qui devait déjà être fonctionnel.

Pour éviter ces effets pernicieux, les pratiques de développement doivent permettre de détecter les défauts le plus tôt possible pendant les activités de réalisation et chercher à réduire l'effort nécessaire pour appliquer les corrections.

9.4.1 Amélioration continue

Atteindre l'état où peu ou pas de défaillances sont rapportées en production peut sembler une utopie. Vous ne pouvez pas vous attendre à ce que cela arrive au premier jour de votre transition à l'agilité, en particulier si le carnet de produit contient au départ plus d'une centaine de requêtes de modifications et rapports de problèmes en plus des items de développement. Cela peut prendre plusieurs mois, voire plusieurs années, avec un dirigeant qui communique régulièrement l'objectif de réduire le plus possible les défaillances en production rapportées par les utilisateurs.

Retour d'expérience : Un objectif clair de qualité

Le président d'une PME de développement logiciel mettait une clause dans ses contrats assurant qu'il corrigerait tout défaut du logiciel à ses frais. Étant une petite société, cela lui conférait un avantage compétitif. Du coup, la qualité du développement a toujours été un enjeu non négociable. Pour que le message soit clair, il disait régulièrement à ses équipes : « *Je ne veux pas voir de défauts en production* ». Puis, il s'assura de soutenir ses équipes afin que cet état se produise.

Leurs applications concernent un domaine d'affaires financier complexe, desquelles on dénombre plus de 1,25 million de lignes de code. Au début de notre collaboration, l'organisation avait dénombré environ 35 défauts par année. En mettant en place l'agilité, en particulier son approche d'amélioration continue, elle a réduit ce nombre à

1. Nous avons vu au chapitre 6 *Gestion de projet* que la vélocité d'une équipe agile se mesure par le nombre de points d'effort terminés au cours d'une itération. Les points d'effort sont une estimation relative de la taille et de la complexité des fonctionnalités faite par l'ensemble des membres de l'équipe pendant les activités de planification.

20 défauts. Sachant qu'elle réalise plus de soixante projets par année – représentant plusieurs centaines de fonctionnalités – et qu'elle déploie en production à toutes les deux semaines, c'est une performance digne de mention.
À chaque fois qu'un défaut est détecté, ce président veille à ce que ce défaut soit apporté à l'ordre du jour de la rétrospective suivante afin d'en identifier la cause et d'améliorer le processus pour qu'il ne se reproduise pas.
Peu d'organisations arrivent à livrer des applications d'envergure, d'une qualité telle que la maintenance ne requiert aucun outil de gestion des défauts. De toutes les sociétés avec lesquelles j'ai travaillé en 26 ans, elle a été la seule à y parvenir.
Sylvie

Principe agile : À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accorde et ajuste son comportement dans ce sens.

Conseil : Faire systématiquement et régulièrement une rétrospective sur la source des défauts de manière à éviter que les mêmes défaillances ne se reproduisent est une manière graduelle d'améliorer les pratiques de développement et la conception de l'architecture logicielle.

Les pratiques de développement

Principe agile : Une attention continue à l'excellence technique et à la qualité de la conception améliore l'agilité.

Votre processus actuel de développement logiciel peut être amélioré en cherchant le moyen de détecter les défauts au plus tôt, si possible au moment même de l'introduction du défaut. Quand le carnet de produit d'une application contient plusieurs centaines ou plusieurs milliers de corrections à appliquer, c'est généralement le résultat d'un processus déficient de développement logiciel, où les pratiques visant l'excellence technique sont absentes ou inefficaces.

Conseil : Les bonnes pratiques telles que le respect de la définition de « terminé », le binômage, l'intégration continue et la méthode de développement piloté par les tests (TDD)[1] peuvent aider en ce sens.

L'architecture logicielle

Lors de sa visite à l'événement Agile Tour Montréal 2010, Ken Schwaber, un des fondateurs de la méthode Scrum, a demandé à l'audience combien de personnes étaient aux prises avec du code difficile à entretenir, qui nuisait au développement logiciel. Après que plusieurs participants aient répondu par l'affirmative, il leur a

1. Traduction de « *Test Driven Development* ».

simplement demandé :
« *Alors, pourquoi l'avez-vous écrit ?* »

Principe agile : La simplicité - l'art de maximiser la quantité de travail à ne pas faire - est essentielle.

Une architecture logicielle peut être un bourbier complexe d'interdépendances avec un couplage élevé insérant des défauts lors de la modification du logiciel.

Dans certains cas, une réingénierie de l'application s'impose car elle évolue difficilement. L'architecture, qui faisait du sens au moment de sa conception initiale, s'est progressivement dégradée et personne ne veut plus la modifier à cause de sa grande complexité. Il est possible de remettre l'architecture sur pied et la pratique de la restructuration du code peut aider les équipes à y parvenir.

Principe agile : Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto-organisent.

Les équipes très performantes sont en mesure de décrire l'architecture de leur application de manière simple et concise. Si ce n'est pas le cas, est-ce que l'architecture a été conçue de façon isolée ? Qu'est-ce qui empêche les équipes de s'approprier le code et la conception de leur produit ?

Les principes des équipes multidisciplinaires, de la conception incrémentale et la pratique du binômage peuvent aider en ce sens[1].

En résumé

Les incidents ne deviennent pas systématiquement des requêtes de modification ou des rapports de problèmes. Parfois, il est possible de corriger les problèmes d'infrastructure et applicatifs sans modifier le logiciel. Malgré tout, il y aura toujours des cas où des requêtes de modification et des rapports de problèmes s'avéreront nécessaires et inévitables.

Afin de bien gérer le flot de requêtes de modification et des rapports de problèmes, il est utile de classifier et bien gérer chaque requête. Il existe quatre grands types de maintenance : perfectif, adaptatif, préventif et correctif. Le choix d'un mécanisme de gestion des requêtes dépend de leur volume et du contexte de l'équipe de développement ou de maintenance. Quatre mécanismes de gestion des requêtes sont proposés :

– La **gestion par le carnet de produit** : lorsque le nombre de requêtes est petit, il est possible de les traiter sans cesser les travaux de réalisation.

– Le « ***SWAT team*** » : des volontaires de l'équipe, à tour de rôle, traitent les requêtes pendant que le reste de leurs collègues continue de développer des items du carnet de produit.

1. Nous vous invitons à revoir le chapitre *4 Architecture incrémentale* pour compléter votre réflexion.

– La **création d'un projet de maintenance** : un inventaire significatif de requêtes justifie la création d'un projet afin de produire une nouvelle version du logiciel.

– Le **Kanban** : lorsque l'équipe (de développement, de maintenance ou *SWAT*) peut difficilement s'engager à livrer un contenu défini dans une itération donnée car des requêtes critiques et importantes affluent de façon imprévisible.

Les défauts résultent typiquement en maintenance corrective. Un grand volume de maintenance corrective peut être le symptôme d'un processus de développement logiciel déficient. Les approches agiles encouragent l'amélioration continue du processus de développement de manière à prévenir les incidents, les défaillances et les défauts. Les pratiques XP et TDD sont conçues pour viser l'excellence technique des équipes de projet.

10

Culture organisationnelle

Objectif

Dans ce chapitre, vous découvrirez les fondements de la culture organisationnelle en tant que point de départ vers une transition agile. Le type de culture de votre organisation pourrait avoir un impact sur la durée de la transition ainsi que sur la taille des défis à surmonter quand le temps sera venu d'adopter des pratiques agiles.

10.1 LES TYPES DE CULTURE

Dans son livre *The Reengineering Alternative* (l'alternative de la réingénierie), William E. Schneider décrit quatre types de culture organisationnelle (Figure 10.1) : *collaboration*, *contrôle*, *compétence et accomplissement*[1]. Le modèle de Schneider est un modèle typologique des cultures organisationnelles de la même façon que le MBTI[2] en est un représentant les types de personnalité des individus. Il ne s'agit pas d'un modèle avec une progression, il est descriptif et, de notre avis, c'est un modèle qui permet d'expliquer de façon plutôt simple le sujet très complexe qu'est la culture organisationnelle.

Une culture peut se définir par les préférences de l'ensemble des comportements adoptés dont découlent des valeurs organisationnelles mises en œuvre au quotidien. Il

1. Le terme « accomplissement » est une traduction libre de l'expression « *cultivation* » utilisée par Schneider.

2. MBTI : « Le Myers Briggs Type Indicator (MBTI) est un test déterminant le type psychologique d'un sujet, suivant une méthode proposée en 1962 par Isabel Briggs Myers et Katherine Cook Briggs. », Wikipedia, http://fr.wikipedia.org/wiki/Myers_Briggs_Type_Indicator .

n'existe pas de bonnes ou de mauvaises cultures et chacune comporte des avantages et des inconvénients, au regard de contextes spécifiques, notamment celui d'une transition agile.

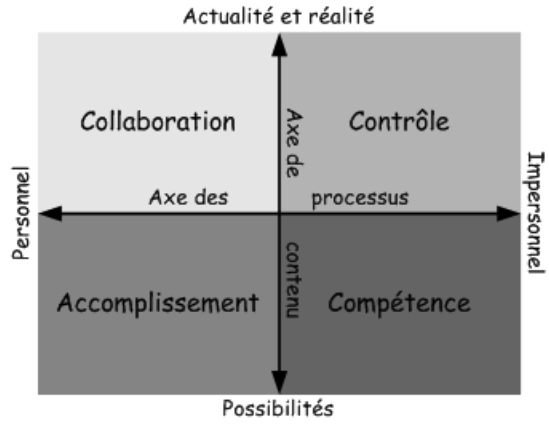


Figure 10.1 — Les quatre types de culture.

Dans son ouvrage, l'auteur explique que chaque organisation n'a qu'une seule culture dominante, à un moment donné dans le temps, reflétant les préférences de l'organisation et de ses dirigeants. Les caractéristiques associées à chaque culture représentent des préférences ou des tendances générales qui ne sont pas exclusives sur toute la ligne. Les cultures dites « pures » sont peu probables, chacune possédant des traits des autres cultures, mais dans une moindre mesure. Ainsi, dans une culture d'accomplissement, des décisions peuvent parfois être basées sur des faits, malgré une préférence à le faire sur des instincts.

Une culture organisationnelle peut être représentée par une surface dont la superficie et la forme dépeignent le degré par lequel les deux axes la définissent (voir la Figure 10.2).

L'axe vertical est celui du **contenu**, ce sur quoi l'organisation est attentive. Au nord de cet axe, les cultures de contrôle et collaboration sont vigilantes à l'actualité et à la réalité, soit ce qui « est ». Au sud, les cultures de compétence et d'accomplissement portent leur attention aux possibilités, soit ce qui « pourrait être ».

L'axe horizontal est celui des **processus**, soit la façon de prendre les décisions et de former les jugements. À gauche, les cultures de collaboration et d'accomplissement ont des processus s'appuyant sur les personnes. À droite les cultures de contrôle et de compétence ont des processus définis et impersonnels.

Chaque culture est issue du mariage unique de contenu et de processus. Cependant, elle ne peut pas se définir par des quadrants diagonalement opposés, le mode de fonctionnement et les valeurs étant incompatibles. Ainsi, la culture de contrôle est opposée à la culture d'accomplissement, et la culture de collaboration est opposée à celle de compétence. Cette notion, importante quand vient le temps de planifier un changement de culture, est discutée à la section 10.4.

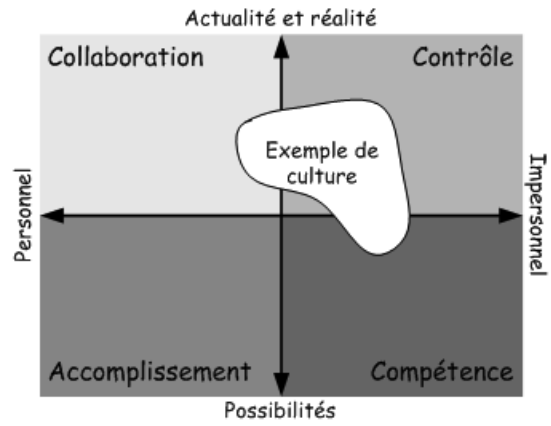


Figure 10.2 — Exemple de représentation d'une culture.

Tableau 10.1 — Caractéristiques de l'axe du contenu identifiées par Schneider.

Actualité et réalité	Possibilités
Réalité concrète et tangible Fondé sur des faits Ce qui s'est passé et ce qui se passe présentement Expériences ou occurrences actuelles Ce qui peut être vu, entendu, touché, pesé, ou mesuré Ce qui est pratique et utile	Idées Alternatives imaginées Ce qui pourrait se produire dans le futur Idéaux et croyances Aspirations et inspirations Nouveautés Innovations, options créatives Concepts théoriques ou cadres de travail Significations et relations sous-jacentes

Il est à noter que certains groupes ou unités d'affaires d'une organisation peuvent avoir une culture qui leur est propre, indépendante de celle du reste de l'organisation. Par exemple, le département de développement logiciel peut avoir une culture différente de celle du département des opérations. Ou encore, une équipe de recherche

Tableau 10.2 — Caractéristiques de l'axe des processus identifiées par Schneider.

Personnel	Impersonnel
Axé sur les gens Organique, évolutif, dynamique Participatif Subjectif Informel Non limitatif Orienté sur ce qui est important pour les personnes Émotif	Détaché Orienté systèmes, politiques et procédures Orienté formules Scientifique Objectif Orienté sur des lois et des principes Formel Sans émotion Prescriptif

et développement pourrait avoir une culture distincte de celle de son organisation. Le mode de fonctionnement isolé ne les empêchera pas d'interagir avec l'organisation.

Les cultures opposées et indépendantes sont des aspects importants lorsque les organisations désirent changer leur culture et nous les traitons plus loin dans le chapitre.

10.2 CONNAÎTRE SON TYPE DE CULTURE

Connaître le type de culture de son organisation permet de mieux gérer les attentes quant à la stratégie d'adoption devant être mise en œuvre et à la durée nécessaire à transition agile. Bien qu'il soit tout à fait possible de mener à bien des projets de développement logiciel en mode agile au sein de chacune de ces cultures, certaines s'y prêtent plus facilement que d'autres.

Schneider décrit les cultures selon cinq aspects :

- **Stratégie** : les critères et la façon dont le succès est évalué.
- **Leadership et gestion** : le style dominant du leadership et des mécanismes de gestion en place.
- **Structure** : comment est structurée l'organisation.
- **Pouvoir et autorité** : les mécanismes et fondements de la prise des décisions.
- **Relations interpersonnelles** : les caractéristiques et comportements des rapports entre les individus.

Le Tableau 10.3 donne l'essentiel des caractéristiques au regard des cinq aspects décrits ci-dessus, en plus d'élaborer sur l'approche générale au changement ainsi que sur des exemples d'organisations de chacun des types de culture.

Au regard du tableau ci-dessus, nous vous invitons à situer la culture de votre organisation avec ces quelques caractéristiques. Si vous n'y arrivez pas nous vous recommandons de lire l'ouvrage de Schneider qui contient un questionnaire vous permettant d'identifier votre culture organisationnelle. Si vous y êtes arrivés, essayez d'imaginer avec quelle facilité ou difficulté vous pourrez mettre en place des pratiques cohérentes avec les quatre valeurs et les douze principes du *Manifeste agile*. Nous tenterons d'anticiper vos questions aux sections suivantes.

Lorsqu'à l'intérieur de ce livre nous parlons de culture et que nous en nommons une en particulier, nous nous référons à l'un des quatre types de culture décrits dans ce présent chapitre. Certaines équipes devront 'choisir leurs batailles' lorsqu'une pratique particulière peut s'avérer être à 'contre-culture', c'est-à-dire dont les valeurs sont incompatibles avec leur culture au point de nécessiter l'adaptation des méthodes agiles au contexte de leur organisation.

Tableau 10.3 — Quelques caractéristiques des cultures selon divers aspects.

Aspect	A : Contrôle	B : Collaboration	C : Compétence	D : Accomplissement
Stratégie	Dominance, être le plus gros, obtenir et garder le contrôle	Synergie, partenariat, nous l'avons fait ensemble	Supériorité, être le meilleur, l'état de l'art	Réaliser son plein potentiel, croître
Leadership et gestion	Autoritaire et directif Maintenir le pouvoir Conservateur et prudent Commandement Ferme, assuré Réaliste	Bâtisseur d'équipes, de confiance et d'engagements Coach Participatif Intégrateur Collégial	Normalisateur Visionnaire Rationnel, analytique Met au défi les subordonnés Pousse les limites Recrute les plus compétents	Catalyseur Semeur, récolteur Inspire et anime les gens Potentialisateur Promoteur, motivateur Appel au partage d'une vision
Structure	Hiérarchie	Amas, groupes	Matrice	Circulaire, réseau
Pouvoir, autorité	Titulaire d'un rôle, d'une position	Axé sur les relations	Expertise	Charisme
Relations interpersonnelles	Ordre Stabilité Objectivité Normalisation Discipline Prédictibilité Utilitaire Réalisme	Égalitaire Diversité Unis, nous sommes forts, divisés nous tombons Implication Pragmatisme Harmonie Spontanéité	Professionnalisme Méritocratie Poursuite de l'excellence Créativité Amélioration continue Efficience Précision Autonomie	Croissance et développement Humanisme Foi en les gens Dévouement, engagement Implication Créativité But Laisser les choses évoluer Viser les étoiles Droit à l'erreur
Approche au changement	Mandate le changement ou lui résiste	Les équipes démontrent de l'ouverture et/ou demandent le changement	Les objectifs de réalisation dictent le changement et ce dernier est bien reçu	Le changement est essentiel et automatiquement accepté
Quête	Certitude et prédictibilité	Unité et connectivité	Liberté d'action, distinction et être unique	Donner un sens, apporter une contribution
Motivation	Pouvoir	Affiliations	Ses réalisations	Se réaliser (en tant qu'individu)
Exemples d'organisations	Fabricants bien établis de produits de base Organisations concernées par des questions de vie ou de mort : Militaire Chirurgie Constructeurs de ponts	Équipes sportives Familles Firmes de consultants Organisations où il y a immédiateté et nécessité d'implication bilatérale : Soins infirmiers Arts et spectacles Services personnels	Organisations offrant des produits ou services de pointe, uniques, distincts, spécialisés et souvent personnalisés Centres de recherche Sociétés d'ingénieurs Universités, collèges et grandes écoles	Où la finalité, la volonté et la réalisation d'idéaux sont ciblées : Organisations religieuses Organisations artistiques Œuvres caritatives Ou encore des produits ou services désignés pour accomplir les desseins d'un ordre supérieur pour leurs clients

10.3 LA CULTURE INFLUENCE LE SUCCÈS DE L'AGILITÉ

10.3.1 Causes des échecs de l'agilité

Il existe plusieurs raisons pour lesquelles un projet agile ne réussit pas. La compagnie VersionOne – une société américaine développant des outils de gestion agile – a publié les résultats de son cinquième sondage sur l'agilité[1]. L'un des points abordés est justement les causes principales d'échec des projets agiles. Selon leur rapport, 22 % des répondants affirment n'avoir jamais connu d'échec de leurs projets agiles. Pour les autres 78 % des répondants, les causes les plus fréquentes d'échec sont données à la Figure 10.3, en ordre décroissant d'importance.

La culture est impliquée dans au moins 19 % des cas, soit la « culture organisationnelle à l'encontre des valeurs et principes agiles » (11 %) et le « manque de transition culturelle » (8 %). De plus, nous soupçonnons qu'une part de ceux ayant répondu « ne sait pas » auraient peut-être donné une raison liée à la culture s'ils y étaient sensibilisés.

Mis à part la culture, notons l'importance à accorder à la formation et l'accompagnement, des enjeux importants qui pourraient réduire jusqu'à 31 % des cas d'échec, soit le « *manque d'expérience avec les méthodes agiles* » (14 %), les « *réticences de l'équipe* » (6 %), « *nouveau pour nous/pas encore complété de projet agile* » (6 %) et la « *formation insuffisante* » (5 %).

À la lumière de ces résultats, y a-t-il une ou des cultures qui se prêtent mieux à l'agilité ? C'est ce que nous voyons ci-après.

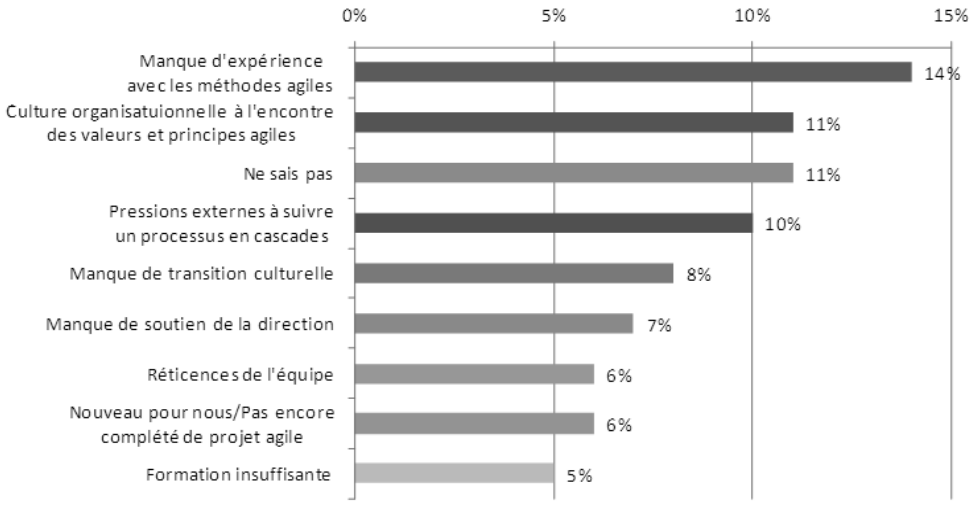


Figure 10.3 — Perception des causes principales d'échecs des projets agiles.

1. Source: VersionOne, State of Agile Survey 2010. Données recueillies auprès de 4 770 participants dans 91 pays, entre le 11 août et le 31 octobre 2010.

10.3.2 La culture de préférence de l'agilité

Certains se sont demandés : *quelle est la culture de préférence pour l'équipe agile idéale ?* Le coach agile Michael Spayd, dans son blog, a mené un sondage en mai 2010[1] afin de trouver des éléments de réponse[2].

Une partie des résultats était attendue, compte tenu de la saveur collaborative des pratiques agiles : 47 % des répondants au sondage perçoivent que la culture de collaboration serait la culture de préférence pour une équipe agile idéale. Un sommaire des résultats est présenté à la Figure 10.4.

Notons aussi que 41 % des répondants considèrent qu'une équipe agile idéale préfère la culture d'accomplissement. L'analyse qui ressort de ce sondage est que l'agilité a une forte propension à être du côté « personnel » sur l'axe horizontal des processus, c'est-à-dire que les mécanismes de décision s'appuient sur les personnes plutôt que sur des procédures définies.

Il ne faut pas conclure qu'il n'est pas possible de réussir à implanter l'agilité dans des cultures de contrôle ou de compétence. C'est possible en adoptant une stratégie adéquate. Cependant, la transition sera plus complexe, plus risquée et probablement plus longue que pour les cultures de collaboration et d'accomplissement.

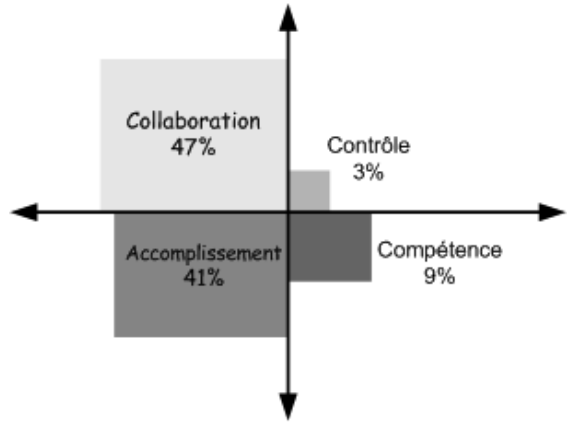


Figure 10.4 — La perception de la culture de préférence pour l'équipe agile idéale, selon un sondage mené auprès d'environ 120 personnes praticiennes de l'agilité.

1. Sondage mené auprès d'environ 120 personnes qui appliquent l'agilité (*Scrum*, *eXtreme Programming* et/ou *Lean-Kanban*) et provenant de cultures différentes.

2. Ce sondage a ses limites. Il faut l'interpréter pour ce qu'il est, soit une perception de ce qu'est la culture de préférence de l'équipe agile idéale, et pas autre chose. Ce n'est pas « la culture actuelle de votre équipe agile » ou « la culture de votre organisation au moment où votre équipe agile a existé ».

10.3.3 Le savoir-être agile

Réussir une transition agile dans une organisation de développement logiciel s'appuie plus fortement sur le savoir-être des individus que sur leur savoir ou leur savoir-faire. Le savoir décrit les connaissances possédées, le savoir-faire décrit les activités appliquées et le savoir-être décrit le comportement à adopter. Ces trois formes de compétences doivent être prises en charge par un programme de formation et d'accompagnement pour éviter des échecs.

L'acquisition et la mise en œuvre du savoir et du savoir-faire agile passent en général très bien dans toutes les cultures, car cela implique d'appliquer les techniques préconisées par les méthodes agiles choisies. Mais cela ne suffit pas. Lorsque des événements inattendus ou non maîtrisés surviennent, c'est le savoir-être qui fait la différence entre un succès et un échec. Et le savoir-être des individus est largement teinté par la culture de leur organisation.

Quand le savoir-être agile n'est pas acquis par les individus, alors ces derniers ont tendance à adapter les pratiques pour qu'elles cadrent dans leur culture. Mais pour adapter de manière adéquate un mode de fonctionnement, il faut comprendre sa raison d'être. Si les compétences se limitent au savoir-faire, les équipes risquent de remplacer une pratique par une autre, incohérente avec la nouvelle philosophie de travail. C'est à ce moment que les individus ont tendance à retourner vers leurs anciennes habitudes plutôt que de s'approprier les nouvelles pratiques, menant parfois à des comportements qui vont à l'encontre de l'agilité.

Lorsque les cultures incitent à l'abandon ou à la modification des pratiques agiles, il faut être vigilant à trouver une alternative répondant aux mêmes valeurs et principes. Par exemple, quand les équipes font la présentation d'éléments ne respectant pas la définition de terminé, le principe de mesurer l'état d'avancement par des incréments de logiciel fonctionnel est escamoté. Du coup, cela a un impact sur le budget, les échéanciers et les risques.

> « *Sans l'évolution de la pensée, la méthodologie devient technique et la pratique devient imitation.* » - Peter Block

10.3.4 Caractéristiques favorables et défavorables des cultures face à l'agilité

Les sous-sections qui suivent mettent en lumière les caractéristiques favorables et défavorables des cultures au regard de l'agilité, notamment lorsqu'une culture donnée est poussée aux limites de la pureté du type. Dans ces cas, ce sont les caractéristiques défavorables de cette culture qui ressortent plus que celles qui sont favorables à l'agilité. Nous présentons nos observations par culture, dans l'ordre décroissant des résultats montrés à la Figure 10.4.

Collaboration

La culture de collaboration prédispose au travail d'équipe. Les individus de cette culture sont souvent de bons généralistes s'attribuant la prochaine tâche d'équipe à faire à pied levé. Et c'est là la principale caractéristique favorable de cette culture pour réussir une transition agile.

Lorsqu'elle est vécue à l'excès, comme les individus veulent tout décider par consensus, soit ils s'embourbent dans des réunions où tous doivent être présents, soit ils cherchent l'avis de tous et de chacun avant de prendre une décision. Des délais sont alors introduits dans le processus. Ce symptôme prend de l'ampleur en proportion de la taille de l'équipe. En effet, obtenir le consensus à quatre personnes est une chose, mais essayer de l'obtenir avec trois équipes de dix personnes sur le même projet, en est une autre. Cette façon de procéder a un impact négatif sur l'efficacité et la productivité.

Conseil : Si le cas se présente dans vos projets, cumulez les données d'effort et de durée passées à obtenir ce consensus et amenez le point à l'ordre du jour de la prochaine rétrospective. Il y a de fortes chances que l'équipe décide de se nommer des représentants pour participer aux décisions au lieu que tous y soient impliqués.

Accomplissement

Dans une culture d'accomplissement, la croissance concerne à la fois les individus et les équipes. En ce sens, les inspections et les rétrospectives sont menées avec un souci d'apprendre, de tirer des leçons et de s'améliorer.

Vécue à l'excès, une organisation de cette culture manque de canalisation envers les résultats. Le flot de nouvelles idées provoque de l'éparpillement, ce qui amène les individus à ne pas terminer le travail entrepris.

Conseil : Mettez l'accent sur une définition de « terminé » envers laquelle chaque membre de l'équipe s'engage.

Aussi, il est fréquent d'y rencontrer des équipes manquant de rigueur et ne produisant pas les indicateurs de performance attendus des méthodes agiles : graphique d'avancement (*burn down*), mise à jour quotidienne de l'effort restant sur les tâches, vélocité, etc.

Conseil : Demandez fréquemment à la personne assumant le rôle de leader de l'équipe (Scrum Master ou autre titre aux responsabilités similaires) de produire ces données jusqu'à ce que l'habitude soit ancrée dans les mœurs des équipes.

Compétence

À l'opposé de la culture de collaboration, une organisation avec une culture de compétence sera majoritairement composée de groupes plutôt que d'équipes. La propension à livrer des produits de qualité supérieure à ceux de la compétition

demande la contribution d'individus spécialisés, ayant de très grandes compétences techniques.

Cependant, cette spécialisation limite le partage du travail entre les individus ayant des compétences spécifiques à leur champ d'expertise. Cet aspect réduit la possibilité de prendre des engagements collectifs.

À l'intérieur d'une organisation avec une culture de compétence excessive, l'engagement d'équipe est absent ou chaotique. Les difficultés rencontrées par un individu spécialisé augmentent la chance qu'il devienne le goulet d'étranglement de ses collègues, eux-mêmes incapables de lui venir en aide.

Mise en situation : Manfred et la base de données

Manfred est le spécialiste de conception et d'administration des bases de données de son équipe. Il est plus efficace que quiconque à concevoir des composantes d'accès aux données qui soient performantes et facilement modifiables.

Son équipe ayant adopté l'agilité récemment, avec des itérations d'une durée d'un mois, Manfred est débordé pour créer les composantes d'accès aux données, travaillant plus de 60 heures par semaine pour ne pas retarder les développeurs qui attendent ses composantes. Il n'a plus de temps pour parfaire la performance des composantes. Parfois, l'équipe n'arrive pas à livrer ses engagements car le client rejette certaines fonctionnalités à cause de leur lenteur. Manfred, dans sa spécialité, est devenu le goulet d'étranglement de son équipe.

Après avoir apporté ce point à leur dernière rétrospective, l'équipe, incluant Manfred, a déterminé que sa spécialité serait mieux utilisée si Manfred accompagnait les développeurs à concevoir les composantes d'accès aux données qu'il aura planifiées. Manfred leur a donné un atelier de quelques heures sur la syntaxe, la sémantique et les autres normes internes à respecter. Ensuite, il a travaillé en binôme avec chacun d'eux afin de favoriser leur montée en compétences. À mesure que les développeurs grandissaient en autonomie, Manfred a fait une revue systématique de leurs composantes développées et il a travaillé avec eux pour améliorer la performance. Au final, tous ont travaillé à un rythme soutenu mais normal et le client n'a plus refusé de fonctionnalités pour des problèmes de performance d'accès aux données.

Dans la mise en situation ci-dessus, l'ajout d'un deuxième spécialiste de base de données pourrait sembler une bonne solution. Même si cela permettait d'achever les ajustements de performance en fin d'itération, le travail serait insuffisant pour occuper les deux individus durant toute la durée des itérations.

Une alternative plus efficace a été d'augmenter les compétences des développeurs, pour qu'ils soient habilités à développer les composantes planifiées par Manfred. Ainsi, l'expertise de Manfred est mise à contribution et l'effort de réalisation est distribué à l'ensemble des membres de l'équipe, permettant de réussir les engagements tout en limitant le temps d'attente des individus.

Conseil : Au moment de planifier chaque itération, assurez-vous de faire participer chaque spécialité qui doit intervenir dans la livraison de chaque fonctionnalité ou

caractéristique afin de budgéter l'effort nécessaire pour toutes les tâches requises. Il sera alors possible d'ajuster la répartition des tâches en fonction de la capacité de chacun (compétences et disponibilité).

Contrôle

Les organisations d'une culture de contrôle sont friandes de faits et de chiffres. Celles-ci font preuve de rigueur et de discipline à propos de la collecte, l'analyse et la présentation des données d'avancement ainsi que les calculs de l'effort restant et de la vélocité.

À l'extrême, une culture de contrôle cultive les secrets, et la politique finit par prendre le dessus sur l'efficacité. La transparence est permise seulement lorsque les résultats plaisent, car l'apparence de succès permet de ménager les susceptibilités. Pour s'assurer de démontrer une progression satisfaisante, la réalité est nuancée. Lorsque c'est la définition de « terminé » qui est escamotée pour des questions de visibilité, l'accumulation d'une « dette technique » s'installe. La transparence et la mesure de l'avancement, prônée par l'agilité, sont alors ébréchées.

Aussi, le concept d'équipe auto-organisée passe mal, pour deux raisons. La première, c'est que les gestionnaires ont l'habitude de décider eux-mêmes et de dire quoi faire aux membres d'équipe ; ils ont souvent un style directif. La deuxième, c'est que les membres d'équipe ne sont tellement pas habitués à décider d'eux-mêmes qu'ils n'arrivent pas à prendre les initiatives qui s'imposent. Ils « retournent dans leurs vieilles pantoufles » en préférant se laisser diriger. S'ils arrivent à livrer leurs engagements, c'est un moindre mal. Mais notre expérience nous indique le contraire. Ce changement de savoir-être demande du temps et de l'accompagnement.

Conseil : Adoptez un style de leadership selon le niveau de maturité des équipes[1].

Le concept d'architecture incrémentale qui émerge d'une équipe auto-organisée ne passe pas bien non plus. La plupart du temps, il y a un architecte occupant une « position » qui lui confère un salaire d'environ le double d'un développeur. Certains pourraient penser que si les développeurs étaient capables de faire de l'architecture logicielle, ils occuperaient tous un poste d'architecte et gagneraient le double de leur salaire. Nous croyons que c'est faux. Un développeur aime développer et il est tout à fait capable de juger de la qualité d'une architecture et d'en proposer des améliorations, sans pour autant vouloir en faire son occupation à plein-temps[2].

La collaboration avec le client peut devenir un réel défi quand l'organisation cultive les secrets. Si quelqu'un veut toujours bien paraître aux yeux du client, il n'aura pas tendance à vouloir l'inclure au sein de l'équipe. Il préférera nommer une personne interne à l'organisation comme « représentant du client ». Cela fonctionne

1. Se référer au chapitre *11 Gouvernance* pour le style de leadership associé à chacun des niveaux de maturité d'une équipe.
2. Nous vous invitons à lire le chapitre *4 Architecture incrémentale* pour plus d'information à ce sujet.

seulement si cette personne maîtrise le domaine d'affaires et les besoins du client à un point où elle prend systématiquement les bonnes décisions relatives au comportement de l'application en développement, ce qui n'est pas le cas bien souvent. Alors l'équipe progresse, mais pas nécessairement dans la bonne direction. De plus, ce représentant du client sera la personne blâmée dès que les défis et les problèmes surviendront.

Conseil : Étant donné qu'il y a plus d'avantages à inclure le client qu'à l'exclure, nous recommandons de vous questionner sur les freins réels de cette pratique et de trouver une solution qui apportera la fluidité dans la communication entre le client et l'équipe. Un client participatif développe rapidement une confiance en l'équipe lui permettant d'être plus conciliant quand les défis et difficultés surviennent.

Dans une culture de contrôle, l'accent est mis sur des processus formels et des faits. À l'extrême, les rétrospectives ne portent que sur les processus et les techniques utilisés et rarement sur les dynamiques humaines ou le savoir-être, qui est souvent évité parce qu'on n'est pas à l'aise avec toute la dimension humaine du travail.

L'agilité aura de la difficulté à survivre dans une organisation répondant à culture de contrôle. Pour réussir son adoption, cette dernière doit clairement comprendre la nécessité d'opérer un changement de culture en profondeur. Quoi que possible, c'est un chantier titanesque pour une grande organisation mue par une culture de contrôle.

Conseil : Lorsque la culture de contrôle pure freine l'adoption de l'agilité, nous vous recommandons de démarrer une cellule indépendante où une culture favorable à l'agilité pourra être incubée. Il faut tabler sur les succès obtenus dans cette cellule pour absorber un nombre grandissant d'équipes, poussant la transition à l'agilité jusqu'aux limites possibles dans l'organisation. La transition complète n'étant pas un objectif pour une organisation avec une culture de contrôle.

10.4 GESTION DE LA TRANSITION AGILE

La culture peut être amenée à changer, notamment avec l'arrivée de nouveaux dirigeants ou avec l'adoption de l'agilité qui apportent des valeurs différentes de celles en place dans l'organisation. Il faut noter qu'un changement de culture à l'échelle d'une organisation demande des mois, voire des années de transition. La durée de cette phase de transition est influencée d'une part par la pression motivant le changement et, d'autre part, par l'adéquation des éléments déployés pour obtenir les résultats souhaités.

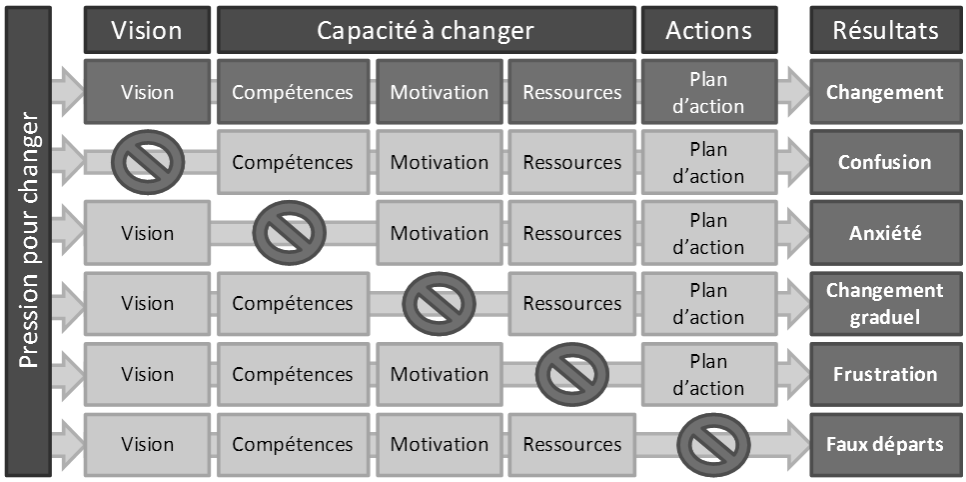
Une pression motivant l'organisation à changer est une condition préalable pour que le changement s'opère à travers la période de transition. Cette motivation provient de deux sources : la **souffrance**, comme source négative, et l'**espoir,** comme source positive. Souvent, ces deux sources sont présentes et s'enchaînent : une situation problématique entraîne de la souffrance et une solution potentielle crée de l'espoir. Ces facteurs de motivation peuvent prendre plusieurs formes :

- **Un sentiment d'urgence** (« *ça passe ou ça casse* ») : la situation actuelle est intolérable et il est impératif de changer rapidement. Ce facteur de motivation influence la durée de la transition parce qu'il impose un changement radical et rapide. La souffrance étant grande, l'organisation devient prête à essayer n'importe quelle alternative pour changer sa situation. Si elle a eu vent du succès de l'agilité dans d'autres organisations, elle sera motivée à le reproduire pour elle-même.
- **Un facteur volontaire positif** : l'organisation a eu quelques résultats mitigés et les équipes de développement désirent améliorer la situation, bien que la souffrance soit faible. Après quelques recherches, il apparaît que l'agilité fait poindre un espoir de combler leurs besoins d'amélioration. Alors l'organisation fait ce qu'il faut pour que les changements s'opèrent adéquatement.
- **Un facteur négatif ou involontaire** : quand quelqu'un – par exemple le client, la maison-mère, ou la haute direction – impose un changement n'étant pas perçu comme nécessaire. Cela crée un inconfort et une souffrance, comme si soudainement une solution se cherchait un problème. À ce moment, il incombe au requérant de bien faire ressortir la problématique menant l'organisation à vouloir adopter l'agilité.

10.4.1 La gestion du changement

Mis à part la pression, cinq éléments sont nécessaires pour qu'un changement se produise, tel qu'illustré à la Figure 10.5. La dernière colonne du graphique donne un aperçu des résultats négatifs obtenus lorsqu'un des éléments est manquant ou inadéquat. Pour prévenir l'échec d'une transition agile, il est important de planifier l'ensemble de ces éléments.

- Une **vision** : La vision doit décrire l'état général ciblé au terme de la transition agile. Elle doit être claire, rassembleuse, communiquée et partagée par tous.
- Des **compétences** : déployer des mécanismes de montée en compétences des personnes impliquées, tels que de la formation, de l'accompagnement, du compagnonnage, des projets pilotes ou de l'expérimentation. Un membre d'équipe compétent pour travailler d'une certaine façon risque de se sentir incompétent pour travailler d'une nouvelle manière. Il faut être attentif à ce que les individus ne tentent pas de retourner dans leur zone de compétences dès que l'occasion se présente. C'est un réflexe normal pour quelqu'un ayant sentiment inconfortable d'incompétence à qui le savoir-faire et le savoir-être nécessaires sont manquants.
- De la **motivation** : chaque individu se demandera « *qu'y a-t-il pour moi dans cette transition agile demandée ?* », « *qu'ai-je à y gagner ?* ». Ce questionnement est légitime et humain. Heureusement, l'agilité étant en vogue, la plupart des personnes qui travaillent dans des projets de développement logiciel sont motivées à l'essayer. Le défi, la nouveauté et le sentiment d'accomplissement qu'apporte l'agilité devraient largement suffire à motiver le personnel. Dans une transition agile, les augmentations de salaire et les bonus sont rarement des



Source : adapté de Dolores Ambrose, 1987. Personal Communication.
Ce modèle provient originalement de « The Enterprise Corporation », une firme de consultants qui a cessé d'exister.

Figure 10.5 — La gestion du changement : éléments menant au succès[a].

a. Source : adapté de Dolores Ambrose, 1987. *Personal Communication*. Ce modèle provient originalement de « *The Enterprise Corporation* », une firme de consultants ayant cessé d'exister.

facteurs de motivation. Par contre, pour maintenir cette motivation, il faut leur fournir les ressources dont ils ont besoin.

- Des **ressources** : dans une transition agile, les ressources sont de la formation, du temps pour expérimenter, des *coaches* ou accompagnateurs, une infrastructure matérielle et logicielle adéquate, la proximité physique et la participation du client, et un environnement de travail qui favorise la collaboration et dans lequel l'équipe est rassemblée[1].
- Un **plan d'action** : un plan décrivant les étapes mettant en œuvre la stratégie. Une stratégie de changement complexe doit s'appuyer sur un plan empirique. Ainsi, plutôt que de tenter d'avoir un plan défini détaillé, il faut mettre en place des outils permettant d'évaluer le niveau d'adoption et de maturité des individus, des équipes, des départements et/ou toute autre entité faisant partie de la portée de la transition agile. Ce plan doit voir à mesurer régulièrement l'état de l'adoption agile, à gérer les risques associés et à ajuster la course en cas de nécessité.

Quand plusieurs éléments sont manquants, les résultats se combinent. Une organisation voulant faire une transition agile doit bien comprendre ce qui constitue la motivation la poussant à changer et s'assurer que cette motivation soit bien

1. C'est-à-dire tous au même endroit physique, bien qu'il soit possible de faire de l'agilité avec des équipes distribuées géographiquement.

communiquée à l'ensemble des personnes impliquées. Cette partie de la démarche initiant le changement est toute aussi importante que les éléments qui la mettent en œuvre.

10.4.2 L'équipe et son nouveau contexte opérationnel

L'agilité ne peut pas exister sans travail d'équipe. Et pour avoir une équipe, il ne suffit pas de mettre ensemble un groupe de personnes en espérant que « ça colle ». Alors qu'entend-on par « équipe » ? Voici ce que nous utilisons pour différencier un groupe d'une équipe :

> « *Un **groupe**, c'est quand chacun de ses membres se contente de faire les tâches qu'on lui attribue ou celles dont il a envie. Une **équipe**, c'est quand tous ses membres sont **conjointement** et **solidairement responsables** de l'atteinte des objectifs auxquels ils se sont engagés.* » - Sylvie[1]

De ces deux énoncés, nous observons les comportements suivants :

- Lorsque des individus justifient leur contribution individuelle dans un contexte d'échec collectif, ils forment un **groupe**. Certains arguments sont des indicateurs du manque de cohésion pouvant exister à l'intérieur d'un groupe :
  - « Bien moi j'ai fait ce que j'avais à faire » ;
  - « J'ai fait ma part, [ce sont les autres qui n'ont pas fait la leur] » ;
  - « J'ai fini mon itération avant les autres » ;
  - « Je n'ai pas pu travailler car j'attendais la livraison d'un autre collègue » ;

Ces arguments ne sont pas ceux d'une **équipe** valorisant la collaboration et la réalisation en collectivité :

- Quand les individus appliquent un savoir-être de membre d'équipe, il est possible d'entendre : « *J'ai fini ce que je devais faire, alors qui puis-je aider ?* ».

La clé d'un changement de culture est de créer un nouveau contexte pour permettre aux individus d'y évoluer pendant leur période de montée en compétences et, surtout, leur donnant le droit à l'erreur. En effet, une personne a moins peur d'essayer de nouvelles pratiques et de nouveaux comportements si son contexte lui donne le droit de se tromper, un *feedback* constructif sur ses erreurs, et de l'aide pour corriger le tout. Le défi est de faire en sorte que les individus intègrent ce nouveau contexte afin qu'ils démontrent un savoir-être de membre d'équipe.

Dans un changement vers l'agilité, l'équipe – en l'occurrence, un petit nombre de personnes – est l'endroit le plus propice pour créer ce nouveau contexte qui peut être assimilé par les individus. En parallèle, les individus doivent être soutenus dans le changement par des gestionnaires leur fournissant les ressources dont ils ont besoin pour évoluer dans cette transition.

1. Définition adaptée d'un atelier de formation d'équipe auquel j'ai participé en 1997.

10.4.3 Élaboration d'une stratégie

Une stratégie de mise en œuvre de l'agilité doit tenir compte de la culture pour réussir.

Conseil : Il est plus facile de démarrer un projet agile du bon pied que de redresser une situation problématique. Alors il est recommandé de se faire accompagner pour diagnostiquer l'état de préparation de votre organisation à adopter l'agilité, incluant les risques liés à votre culture organisationnelle. Ensuite, il faut élaborer une stratégie de transition tenant compte des résultats du diagnostic.). Sinon, les changements apparents ne seront pas pérennes et le risque d'échec sera élevé.

Lorsqu'un nouveau dirigeant veut amorcer un changement culturel en profondeur, il doit être conscient qu'il ne sera pas possible de passer directement d'un type de culture donnée vers le type de culture opposé. Cette notion devient importante car pour réussir un changement de culture de 180°, la stratégie doit immanquablement prévoir de cheminer un moment à travers l'une des cultures adjacentes.

Conseil : Il est plus facile de démarrer un projet agile du bon pied que de redresser une situation problématique. Alors il est recommandé de se faire accompagner pour diagnostiquer l'état de préparation de votre organisation à adopter l'agilité, incluant les risques liés à votre culture organisationnelle. Ensuite, il faut élaborer une stratégie de transition tenant compte des résultats du diagnostic.

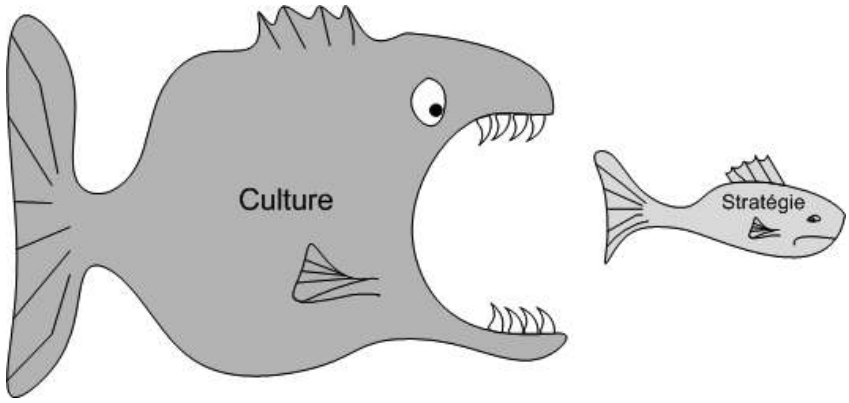


Figure 10.6 — « La culture mange de la stratégie au petit-déjeuner » – Peter Drucker

En résumé

Adopter l'agilité à l'échelle organisationnelle requiert une période de transition dont la durée dépendra de la culture organisationnelle présente au début de la transition. Nous avons montré quatre types de culture avec leurs caractéristiques : *contrôle, collaboration, compétence et accomplissement*. Chacune de ces cultures a des caractéristiques favorables et défavorables face à l'adoption de l'agilité.

Pour qu'une organisation réussisse sa transition à l'agilité, elle doit opter pour une stratégie tenant compte de sa culture afin de tabler sur les caractéristiques favorables et d'atténuer l'impact de ses caractéristiques défavorables.

11

Gouvernance

Objectif

Dans ce chapitre, vous découvrirez l'essentiel des impacts de l'adoption de l'agilité sur les objets de gouvernance et sur le rôle des gestionnaires. Vous verrez comment ce rôle demeure au premier plan pour soutenir les équipes autogérées et atteindre les objectifs établis.

Mise en situation : Charles et la transition à l'agilité, la suite...

(Suite de la mise en situation du chapitre *1 Qu'est-ce que l'agilité ?*, où Charles est le directeur des TI d'une organisation gouvernementale).

Charles est un des participants du comité exécutif qui réunit tous les cadres du même niveau hiérarchique que lui, en plus du directeur général. Le comité exécutif établit les plans stratégiques triennaux, fixe les objectifs annuels et assure le suivi à haut niveau des budgets, des échéanciers et de tous les paramètres permettant de voir si les objectifs sont en bonne voie d'être atteints.

Au cours des dernières années, les besoins d'affaires ont évolué et les applications sont devenues graduellement désuètes. L'organisation doit en faire une refonte complète car les technologies actuelles ne permettent plus de répondre de manière adéquate aux nouveaux besoins d'affaires. Le budget requis s'annonce énorme et les risques de dépassement sont élevés. La stratégie est de rapprocher les affaires des équipes de développement afin de se doter d'applications utiles avec beaucoup de valeur d'affaires dès les premières livraisons. Compte tenu que les processus de développement sont aussi à refaire, le comité exécutif profite de l'occasion pour moderniser ses façons de faire en adoptant la méthode Scrum[1], introduisant ainsi un cycle de vie des projets incrémental et itératif.

1. Pour plus d'informations sur les méthodes agiles, se référer au chapitre *2 Les méthodes agiles*.

Le plan stratégique comprend trois phases :
1 – Exploration par quelques projets pilotes (six mois) ;
2 – Déploiement sur une sélection de projets (huit mois) ;
3 – Institutionnalisation (deux ans).
De cette stratégie, découle un premier plan directeur prévoyant la mise sur pied d'un centre d'excellence pour soutenir les équipes impliquées dans les projets pilotes et assurer le suivi du plan directeur auprès du comité exécutif. Ce centre d'excellence permet de catalyser la transition agile, autant dans les équipes qu'auprès des gestionnaires, pour déployer efficacement la stratégie de l'organisation.
Au cours des premières activités de la phase d'exploration, les membres du comité exécutif constatent qu'ils doivent revoir la structure hiérarchique parce que les rôles changent, et celle de gouvernance parce que la structure actuelle ne convient plus à assurer l'atteinte d'objectifs complètement transformés.

11.1 AGILITÉ ET ALIGNEMENT STRATÉGIQUE

L'adoption de l'agilité entraîne un lot de changements et de risques au niveau organisationnel. C'est pourquoi nous recommandons que les gestionnaires aient une position claire et des objectifs partagés au sujet de l'agilité. Le plaisir de changer et la réponse à un effet de mode ne peuvent pas justifier une transition agile. Ce sont les besoins d'affaires connus et partagés qui justifient un tel changement.

11.1.1 Compréhension commune de l'agilité

Lorsque l'organisation choisit l'agilité pour des raisons stratégiques, il est important que tous ceux ayant un rôle à jouer dans le soutien de cette initiative en deviennent les promoteurs. La nature du soutien attendu par la direction est importante, car le seul financement de la transition n'est pas suffisant. L'agilité est une philosophie de travail et il est possible que l'organisation doive réunir certaines conditions gagnantes, qu'elle accepte de se faire bousculer et qu'elle soit réceptive à comprendre les raisons et les objectifs du manifeste, des méthodes et des pratiques agiles. Cela est possible s'il existe un consensus sur les raisons ayant introduit l'agilité en tant que solution d'affaires et un engagement à faire un succès de l'initiative.

Chaque gestionnaire peut en avoir une opinion ou une compréhension différente d'un autre. Certains en ont peut-être des préjugés défavorables. Avant de solliciter leur soutien à une transition, il est impératif que les dirigeants d'une même organisation aient une définition commune de **ce qu'est** et de **ce que n'est pas** l'agilité.

Afin de découvrir comment l'agilité peut être désignée comme enjeu stratégique, voyons à travers ses quatre piliers ce que les dirigeants peuvent attendre de sa mise en œuvre :

1. **Un processus empirique** visant à éliminer le gaspillage, à améliorer l'efficience et la productivité et ce, itération après itération pendant toute la durée des projets.

2. **Un carnet de produit priorisé** pour concentrer la planification du développement sur les fonctionnalités ayant la plus grande valeur d'affaires. Cela permet de répondre aux opportunités et aux risques survenant durant les projets, ainsi que d'économiser sur les efforts gaspillés au développement de fonctionnalités peu ou pas utiles[1]. Ce carnet permet de tirer le meilleur bénéfice possible des ressources engagées. En conséquence, ce pilier offre une meilleure information quand vient le temps de planifier à l'intérieur de contraintes de coûts et de délais.
3. **La livraison d'incréments de produits terminés** pour mesurer l'avancement des projets et la livraison de valeur de façon fiable, fondé sur la livraison périodique de fonctionnalités complètes, sans concession à la qualité. Cela permet de réduire l'incertitude et la criticité des périodes de stabilisations en fin de projets.
4. **Une équipe auto-organisée** pour motiver et engager tous les intervenants impliqués et soulager les goulets d'étranglements que sont souvent devenus les spécialistes de l'organisation. Chaque équipe est un collectif de cerveaux visant l'objectif de livrer le maximum de valeur de la manière la plus efficace possible.

Ces piliers offrent des avantages et des bienfaits recherchés par les organisations évoluant dans un contexte compétitif, là où leurs méthodes actuelles ont parfois atteint les limites de leur performance. Ces avantages et bienfaits sont aussi appréciés, voire nécessaires, dans des contextes moins compétitifs où les gains d'efficacité et les réductions de coûts sont ciblés.

L'agilité est une approche ou une philosophie de travail mise en œuvre par un ensemble de méthodes, pratiques et techniques comme Scrum, Kanban, *eXtreme Programming*, TDD, DSDM, Crystal, *Agile UP*, etc. Bien qu'elles soient dites légères, ces méthodes requièrent de la **rigueur** et de la **discipline**. L'agilité n'est pas une excuse au manque de méthode de travail, au chaos et à l'indiscipline et elle ne signifie pas que :

- vous ne ferez plus de gestion de projet : la planification, le suivi et la gestion de risques se font différemment en agile ;
- vous ne produirez plus de documentation : le client décide, avec l'équipe, quels documents sont utiles et nécessaires ;
- vous abandonnerez les équipes à leur sort : l'autogestion d'une équipe requiert du soutien de la part des gestionnaires ;
- vous ferez les choses à moitié : les conditions de succès et la définition de « *terminé* » établissent les critères de qualité ;
- vous changerez toute la portée des projets du jour au lendemain : même si le changement est le bienvenu, il faut une cible pour éviter de s'éparpiller et de développer n'importe quoi ;

1. Rappelons ici la troublante statistique du Standish Group à l'effet que 45% des fonctionnalités développées ne sont jamais utilisées.

- vous n'aurez plus de problèmes : l'agilité rend les problèmes visibles mais ne les résout pas. C'est à l'organisation qu'incombe de les traiter de manière adéquate.

Avec une compréhension commune, il est plus facile de s'entendre sur les répercussions possibles et collaborer sur un plan de mise en œuvre adéquat.

11.2 IDENTIFIER LES RÉPERCUSSIONS SUR L'ORGANISATION

11.2.1 Répercussions sur les objets de gouvernance

Une organisation ayant choisi d'adopter l'agilité doit tenir compte des effets d'une telle adoption sur ses différents objets de gouvernance. Le Tableau 11.1 montre les raisons et le niveau d'impact sur les objets typiques de la gouvernance.

Tableau 11.1 — Niveau d'impact de l'adoption de l'agilité l'agilité sur les objets de gouvernance.

Objets de gouvernance	Niveau d'impact	Raisons
Les affaires	Élevé	Lorsque les affaires ne sont pas impliquées directement dans le projet de développement, il arrive qu'elles reprochent aux TI de ne pas avoir correctement répondu à leurs demandes. En adoptant l'agilité, les affaires doivent s'impliquer activement dans les projets de développement, allant même jusqu'à affecter un responsable de produit à plein-temps au sein de l'équipe de projet. Ce changement majeur implique des réaffectations, temporaires ou permanentes dans des postes clés aux affaires. Il est aussi possible de devoir réaligner les nouvelles capacités de livraison des solutions avec la capacité des unités d'affaires à accueillir les changements de solutions à une fréquence plus élevée.
Les TI	Élevé	L'interaction et la collaboration des TI avec les affaires seront impactées. Les processus, les pratiques, les rôles et responsabilités seront changés.
La gestion de projets	Élevé	La reddition de compte est particulièrement impactée par l'adoption de l'agilité et les nouveaux mécanismes vont bousculer les habitudes de nombreuses personnes : équipes, chargés de projets, bureau de projets, gestionnaires et toute personne impliquée dans le suivi des projets. Aussi, le modèle d'affectation du personnel risque d'être grandement impacté. L'affectation et le suivi des budgets en fonction des priorités d'affaires risquent de changer également, en plus de la comptabilisation des coûts de projets.

Tableau 11.1 — (suite)

Les rôles et responsabilités	Moyen	Certains nouveaux rôles seront créés, d'autres seront transformés, notamment dans le secteur des TI et celui des affaires où un responsable de produit doit être défini. Le rôle des gestionnaires est aussi transformé, dans un contexte où les équipes matures et autogérées sont favorisées, déplaçant ainsi une partie des responsabilités des gestionnaires vers les équipes.
La gestion du changement	Moyen	L'adoption de l'agilité étant un changement organisationnel important, les mécanismes existants de gestion du changement seront sollicités pour toute la durée de la transition (de plusieurs mois à plusieurs années, selon la taille de l'organisation et l'urgence perçue de changer).
Les risques opérationnels	Faible ou élevé	L'agilité aura peu d'influence sur la façon dont les risques opérationnels sont gérés, à moins que toute l'organisation soit dans le domaine du développement logiciel. Dans ce cas, l'agilité aura un impact important sur la majeure partie des opérations en transformant les façons de faire de tous.
La conformité (aux lois, règlements et normes applicables)	Faible	Les mécanismes visant à assurer la conformité aux lois et règlements ne devraient pas changer avec l'adoption de l'agilité. Un peu de changement pourrait survenir dans ce qui a trait à la conformité à des normes (ex. ISO 9001) et modèles (ex. CMMI) principalement à cause des changements de rôles et de certaines pratiques. Mais l'impact se limiterait à revoir la matrice de traçabilité entre les exigences de telles normes et les nouvelles pratiques. La documentation réglementaire ou normative doit continuer d'être produite en tant qu'exigences d'affaires, agilité ou pas.
La sécurité	Faible	Le domaine de la sécurité est vaste : personnes, biens immobiliers et autres actifs, infrastructure et information. L'agilité change surtout la séquence et les pratiques de travail, mais peu ou pas les exigences liées à la sécurité, ni à la façon dont elle est mise en œuvre dans les applications.

À la lumière de ces informations, il est clair que les plans directeurs doivent adopter des éléments tactiques spécifiques pour profiter des bienfaits de l'agilité et gérer les enjeux ainsi que les impacts sur les objets de gouvernance. Ensuite, pour opérationnaliser les transformations, l'organisation doit choisir sa stratégie d'adoption (voir la section *Mise en œuvre de l'agilité*).

11.3 LE POSITIONNEMENT DE L'IMPUTABILITÉ VERS LES AFFAIRES

L'adoption de l'agilité n'est pas et ce ne doit pas seulement être une initiative des TI, car il serait difficile d'appliquer les valeurs et les principes agiles efficacement sans impliquer les affaires au cœur des projets de développement logiciel. Les TI

ne sont donc plus les seules imputables du succès des projets. S'il existait une relation client-fournisseur entre les affaires et les TI, celle-ci doit être transformée en relation de partenariat pour assurer le succès de la démarche. Et ce peu importe si le développement logiciel se fait au sein de l'organisation ou en sous-traitance.

Dans cette transformation, il importe de considérer ces deux parties prenantes actives – affaires et TI – comme ayant chacune leur part d'imputabilité. Les affaires sont typiquement imputables :

- de soutenir les initiatives du responsable de produit ;
- d'éliminer les obstacles rencontrés par les équipes ;
- d'orchestrer les activités autour de la livraison : déploiement, formation, marketing, etc. ;
- d'assurer le succès de chacun des projets d'affaires impliquant du développement logiciel.

Du côté des TI, il est attendu qu'elles soient imputables d'assurer :

- la qualité des logiciels livrés ;
- l'amélioration continue des processus, méthodes et outils de développement sur les axes de coûts, de délais et de qualité ;
- l'amélioration de la cohésion des équipes ;
- la montée en compétences des membres d'équipes.

Cet enjeu de positionnement de l'imputabilité comporte des risques devant être atténués :

1. **Les responsables d'affaires refusent de changer ou de participer** : en général, c'est parce qu'ils ne comprennent pas ce qu'est l'agilité ni ce qu'elle apporte. Peut-être aussi qu'ils n'ont pas la capacité de recevoir des livraisons fréquentes, aspect souvent sous-estimé des équipes de projet lorsque les affaires participent peu en leur sein. Ce risque s'atténue avec de la formation et du coaching. Mais il faut aussi inclure les avantages et les bienfaits recherchés de l'agilité en tant qu'objectifs dans le plan stratégique – par exemple, livrer fréquemment, livrer plus de valeur d'affaires, livrer plus de qualité, réduire les coûts de développement[1].
2. **Les TI ne veulent pas se rendre jusque-là parce qu'ils perdent du pouvoir** (budget et imputabilité). S'ils sont les seuls responsables des projets, alors ils ont le droit de dire *non*, ce qui va à l'encontre du succès dans bien des cas. Cette réaction, pourtant réelle, est paradoxale car l'agilité est habituellement introduite par les TI. Ce risque s'atténue en faisant participer les TI à l'élaboration des enjeux et des objectifs d'affaires. Cela implique souvent de changer le modèle de gouvernance : nouveau plan stratégique, coaching et formation pour faire accepter le changement et monter en compétences les gestionnaires impliqués dans leur nouveau rôle.

1. Se référer aux avantages et aux bienfaits de l'agilité au chapitre *1 Qu'est-ce que l'agilité ?*

Ces risques et leurs activités d'atténuation nous amènent à être sensibilisés à la manière dont il est préférable d'approcher la direction afin d'aligner les motivations de tous.

11.4 LES RÉPERCUSSIONS SUR LE RÔLE DE GESTIONNAIRE

Il est possible que leurs premiers contacts avec l'agilité insécurisent certains dirigeants craignant de perdre leurs acquis : contrôle, pouvoir et avantages liés à leur titre. Avant de solliciter leur appui, les dirigeants ont besoin de comprendre comment leur rôle sera transformé et quels bénéfices ils pourront en tirer. En d'autres termes, ils ont besoin de répondre aux questions *qu'arrivera-t-il à mon poste ?* et *qu'y a-t-il pour moi dans l'agilité?*

Semblable aux transformations du rôle de chargé de projet[1], le rôle de gestionnaire, ou de cadre intermédiaire, doit fort probablement être transformé. De gestionnaires de personnes, de tâches ou de projets, ils deviendront des gestionnaires d'objectifs et de contraintes[2] :

- définir les objectifs de leurs équipes ou leurs départements au lieu de mettre l'accent sur leurs tâches ;
- prioriser les objectifs au lieu de prioriser les activités ;
- suivre la progression vers l'atteinte de ces objectifs ;
- continuer de protéger les ressources des équipes afin d'augmenter leur cohésion et, par conséquent, leur performance ;
- soutenir les employés dans leur développement de carrière ;
- construire et entretenir des relations avec les autres départements et équipes.

Le rôle de gestionnaire demeure donc au premier plan d'une transition agile. Comme pour les autres rôles de l'organisation, cette transformation demande du temps et se fait graduellement, au gré de la maturité de l'organisation. Au début de la transition, les gestionnaires auront probablement besoin de coaching pour saisir pleinement le changement de philosophie et faire face aux objections qui feront surface, ayant pour conséquence de ralentir la transition.

Anticiper les objections

Tout changement amène son lot de résistance et les gestionnaires ne font pas exception. Une mauvaise compréhension entraîne nécessairement des objections.

1. Ce sujet est discuté au chapitre 6 *Gestion de projet*, dans la section des répercussions sur les intervenants.
2. Traduit et adapté d'un billet de Martin Proulx, président de Pyxis Technologies, *I dont't feel so good – I'm a people manager in an agile organization*, http://analytical-mind.com/2010/08/12/mommy-i-dont-feel-so-good-im-a-people-manager-in-an-agile-organization/, consulté en avril 2011.

> « *Oui mais, avec l'agilité, je vais perdre le contrôle puisque les équipes seront autogérées !* »
> « *Mais comment va-t-on y arriver ? Nous manquons déjà de temps pour tout faire...* »
> « *Mais pourquoi l'agilité ? Le statu quo n'est pas si mal que ça...* »
> « *Ah mais c'est que j'ai des intérêts personnels et l'agilité ne me sert absolument pas !* »
> « *Là, on me demande de faire appliquer une nouvelle méthodologie à mes équipes et personne ne m'a impliqué dans la décision.* »

Ces quelques réactions, nous les avons entendues à maintes reprises et nous les considérons normales, voire prévisibles quand la transition agile est embryonnaire. Elles émanent typiquement de la peur de perdre quelque chose.

Pour prévenir ce type de réaction, Mike Cohn dans son ouvrage *Succeeding with Agile*, propose d'identifier :

- les personnes risquant de perdre quelque chose si la transition agile devient un succès ;
- les coalitions pouvant se former et s'opposer à la transition agile.

Cette liste permet d'établir un plan d'action ciblé sur les individus visant à augmenter leur adhésion au changement avec des interventions comme de la formation, de l'accompagnement et la définition de leur implication.

Aussi, il est fort probable qu'il faille réviser les éléments de rémunération afin de favoriser leur motivation face aux nouveaux éléments du plan stratégique et des objectifs qui en découlent.

11.5 LES RÉPERCUSSIONS SUR LES STRUCTURES EN PLACE

L'agilité peut avoir un impact significatif sur les structures organisationnelles et de gouvernance s'expliquant en partie par le rapprochement entre les unités d'affaires et celles des TI, et en partie par les changements d'imputabilité.

Il est possible qu'un processus à la chaîne soit installé au fil du temps dans la structure, créant un éloignement entre les utilisateurs des systèmes et les équipes de développement. Cette chaîne peut, par exemple, se composer des *pilotes d'applications*, des *utilisateurs experts*, des analystes d'affaires, des architectes d'affaires, des chargés de projet, des architectes de logiciel et des concepteurs. Cette situation donne un carcan structurel à ces organisations dont l'organigramme est calqué sur leur processus traditionnel. En changeant de processus, ces organisations peuvent avoir de la difficulté à s'adapter car cette structure est très ancrée dans leurs mœurs, affectant les interactions entre les individus, leur contrat de travail, leur rémunération et leurs possibilités d'avancement.

Les approches agiles proposent de faire interagir et contribuer les intervenants ensemble, plutôt qu'ils soient sollicités à la chaîne. Cela débute par les responsables d'affaires qui devront choisir un responsable de produit pour s'impliquer activement

dans les projets de développement. Les gestionnaires doivent comprendre l'impact que la révision des différents rôles des intervenants peut avoir non seulement sur la structure organisationnelle mais aussi sur celle de gouvernance.

11.5.1 Répercussion sur la structure organisationnelle

Les gestionnaires se dotent de certains outils afin de connaître, définir et diffuser la structure organisationnelle envisagée comme le moyen de mise en œuvre de la mission. Par exemple, l'organigramme, permettant d'illustrer la répartition des membres de l'organisation selon leurs responsabilités et leurs liens hiérarchiques. Bien que ce diagramme permette d'identifier le titre des individus et leurs liens de contrôle et de subordination, il ne permet pas de voir la nature des responsabilités de chaque rôle de l'organisation. Un autre outil, le RACI[1], permet de catégoriser sous quatre types le niveau de responsabilité des intervenants concernés par chacun des livrables et des activités de l'entreprise :

- Le **R**esponsable, dont le rôle est d'assurer la réalisation d'une activité ou de la production d'un livrable. Il est à noter qu'il est avant tout le responsable du résultat et que l'exécution peut être partagée ou déléguée à un autre individu.
- L'**A**pprobateur, dont le rôle est d'approuver que le contenu soit adéquat. Dans le cas contraire, l'approbateur peut appliquer un droit de veto en demandant au responsable des résultats de corriger la non-conformité, car il est imputable du résultat.
- Les intervenants **C**onsultés, dont le rôle est de rester disponibles car ils sont susceptibles de contribuer à la tenue d'une activité et/ou au contenu du livrable.
- Les intervenants **I**nformés, à qui le résultat de l'activité ou le livrable sera communiqué, suite à son approbation.

Le Tableau 11.2 illustre une organisation où la hiérarchie occupe une grande part de responsabilité dans l'approbation des livrables de l'organisation. Il y a de fortes chances pour que l'introduction de l'agilité amène le gestionnaire d'une telle organisation à revoir le RACI à la lumière des valeurs, principes et méthodes nouvellement adoptées. La formation d'équipes autogérées permet de croire que le rôle de l'autorité hiérarchique, tel que représenté dans ce tableau, n'est plus le meilleur choix d'approbateur. Probablement que le destinataire du livrable est l'individu le mieux qualifié pour valider le contenu d'un livrable qu'une figure d'autorité. De la même façon, la responsabilité d'approuver les plans de projets et les comptes-rendus d'avancement se déplace du directeur vers le responsable de produit affecté au projet.

1. Source : Le guide du corpus des connaissances en management de projet (Guide PMBOK) du Project Management Institute, 4e édition, 2008.

Tableau 11.2 — Exemple de matrice RACI dans un contexte non agile.

Rôles / Responsabilités (activités et livrables)	Vice-président	Directeur	Chargé de projet	Architecte	Analyste	Développeur
Plan stratégique	A	R	C	C	I	I
Plan de projet	I	A	R	C	C	I
Dossier de conception			A	R	C	I
Dossier d'analyse		I	C	A	R	I
Compte-rendu d'avancement	I	A	R	C	C	C

À la différence d'un chargé de projet ayant souvent un rôle d'autorité, le coach d'équipe (appelé Scrum Master dans la méthode Scrum) est responsable d'encadrer l'équipe en la guidant et en la faisant évoluer. Il est certes un contributeur important mais il n'a pas d'autorité hiérarchique sur une équipe autogérée. Cette dernière devient responsable d'un ensemble d'activités et de livrables issus du processus de développement et elle doit s'auto-organiser pour livrer ce qui a été convenu avec le responsable de produit.

La révision du RACI permet d'illustrer comment l'adoption de l'agilité cause une transformation à la structure organisationnelle. Le Tableau 11.3 illustre la transformation du précédent cas au regard des transformations décrites ci-dessus.

Tableau 11.3 — Exemple de matrice RACI dans un contexte agile.

Responsabilités (activités et livrables) / Rôles	Vice-président	Directeur	Client	Coach d'équipe	Équipe
Plan stratégique	A	R		C	I
Plan de projet	I	I	A	C	R
Dossier de conception			C	C	R/A
Dossier d'analyse			A	C	R
Compte-rendu d'avancement	I	I	A	C	R

L'agilité risque d'apporter de nouveaux rôles et responsabilités. Si cela s'avère lourd à changer d'un point de vue de la gestion des ressources humaines, alors les titres existants pourraient demeurer les mêmes dans un premier temps. Une distinction doit cependant être faite entre le titre d'une personne et l'ensemble des rôles et responsabilités que ses compétences lui permettent d'accomplir.

Conseil : Si la structure organisationnelle et le processus sont fortement liés, il est recommandé de viser leur indépendance au moment de concevoir des modifications à l'un ou l'autre. Cela apporte nettement plus de flexibilité et de vitesse de réaction quand vient le temps de les améliorer.

11.5.2 Répercussion sur la structure de gouvernance

La structure de gouvernance décrit l'ensemble des individus, groupes, entités administratives ou comités responsables des prises de décision à différents niveaux. Dans les petites organisations où ces décisions sont prises par les individus qui occupent des positions hiérarchiques, il est possible que la structure organisationnelle et celle de gouvernance soit la même. Dans les moyennes et grandes organisations, elle est typiquement parallèle à la structure organisationnelle. La structure de gouvernance est alors constituée de comités dont la composition requiert quelques cadres provenant de la hiérarchie mais surtout d'experts compétents au regard de l'objet de gouvernance dont ils sont imputables.

Il est possible que le plan d'affaires ou le plan stratégique ait porté sur les mêmes objets de gouvernance depuis des années, voire des décennies, et que seules les valeurs des objectifs aient changé au fil du temps. Dans ce cas, la structure de gouvernance s'est peut-être installée de façon rigide. En adoptant l'agilité, il est fort probable que les stratégies amènent l'organisation à revoir sa structure de gouvernance et la composition de ses divers comités.

Comme pour la structure organisationnelle, les comités transformés ou créés doivent apparaître dans le RACI de la structure de gouvernance dont les activités et livrables découlent des objets de gouvernance. Ces comités apparaîtront aussi dans les RACI opérationnels, là où ils doivent contribuer, approuver ou être tenus informés.

Les dirigeants doivent comprendre qu'une transformation agile pourrait avoir un impact significatif sur leur structure de gouvernance. La flexibilité ou la rigidité de la structure de gouvernance en place est un facteur déterminant du temps requis pour la transformation et les plans directeurs doivent en tenir compte pour assurer le succès de la démarche.

11.5.3 La culture : levier ou frein à l'adoption de l'agilité ?

Tandis que certaines cultures adoptent l'agilité avec aisance, (les cultures de *collaboration* et d'*accomplissement)*, d'autres mettent en théorie plus de temps pour y parvenir, (les cultures de *compétence* et de *contrôle)*. Pour réussir l'adoption de l'agilité, la culture de départ doit être analysée pour identifier les nouveaux comportements spécifiques à adopter. Cette culture est en premier lieu reflétée par les gestionnaires dont les comportements envoient des messages clairs sur ce qu'ils valorisent. En adoptant les valeurs agiles, les gestionnaires devront immanquablement modifier certains de leurs comportements.

Chaque organisation est un cas particulier à analyser. Toutefois, nous recommandons de s'assurer que les changements souhaités de culture ne soient pas trop éloignés de la culture en place car les changements trop brusques influencent négativement les résultats de l'organisation.

11.6 LA MISE EN ŒUVRE DE L'AGILITÉ

Avec le soutien de la direction et une bonne compréhension des répercussions, les responsables d'affaires et les TI peuvent élaborer un plan tenant compte de leur capacité respective afin de s'assurer que les objectifs soient réalistes et correctement budgétés. De plus, chaque groupe ou unité d'affaires de l'organisation responsable de la mise en œuvre des plans directeurs doit recevoir la formation et l'accompagnement nécessaires à la réussite de la transition. En particulier, les plans directeurs doivent prévoir la formation des gestionnaires, l'implication du bureau de projet, la révision des mesures et indicateurs et la révision des structures organisationnelle et de gouvernance.

11.6.1 La gestion de changement

Dans le chapitre *10 Culture organisationnelle*, nous avons abordé les éléments pour la gestion du changement de manière générale. Plus spécifiquement, nous les transposons ici en critères de succès d'une transformation organisationnelle agile (Tableau 11.4).

Tableau 11.4 — Les critères de succès d'une transition agile.

Les éléments de la gestion du changement	Les critères de succès
1. Pression pour changer	Raisons pour adopter l'agilité.
2. Vision	Compréhension de l'agilité par les dirigeants et les gestionnaires ; Évaluation des impacts sur les pratiques de gouvernance en place ; Vision claire, communiquée et partagée.
3. Capacité à changer :	
a) Compétences[a]	Formation et accompagnement : en plus des membres d'équipe, ne pas négliger les gestionnaires pour qu'ils soutiennent la transition agile ; Champion de l'agilité dans chaque unité d'affaires.
b) Motivation	Équipes autogérées avec une vision d'équipe et des objectifs communs.
c) Ressources	Cadre méthodologique commun dans lequel on instancie les méthodes agiles choisies ; Principes directeurs sur les rôles et pratiques clés ; Outils adéquats pour soutenir les pratiques agiles : doivent permettre le suivi par les gestionnaires et les dirigeants, autant que pour les membres d'équipe.
4. Actions	Plan d'action : Pourquoi, quoi (stratégie et activités), qui, quand, comment, combien.
5. Résultats	Communication des résultats : succès et leçons tirées.

a. Se référer au chapitre *17 Gestion des individus* pour plus d'informations.

Il ne faut pas négliger un critère de succès plus qu'un autre car il y aurait des ratés, de la frustration, de la confusion, des faux départs et autres conséquences indésirables n'apportant pas le succès anticipé.

11.6.2 Gérer les risques de l'adoption de l'agilité

La transition à l'agilité est unique dans chaque organisation, ce qui rend difficile une identification complète des risques liés à cette transition. Pour être en mesure d'identifier ces risques, il importe d'avoir un portrait le plus exhaustif possible des éléments impactés par une transition agile. Étant donné la complexité d'une transition agile, il sera important de réévaluer ces risques durant la transition. L'ensemble de ce livre vise justement à outiller quiconque doit planifier et gérer une transition agile à l'identification des impacts. Une fois ces impacts identifiés, il devient plus facile, en tenant compte de la culture et du contexte organisationnel d'identifier les risques liés à la transition agile, puis de planifier les mesures d'atténuation et de contingence ressortant de leur analyse.

L'autre difficulté importante est d'appliquer les mesures appropriées d'atténuation et de contingence des risques. Ces deux difficultés font en sorte qu'il est primordial d'accorder une attention particulière à la mise en œuvre de l'agilité nécessitant une série d'activités, dont :

- **Évaluer les impacts** d'une transition agile en tenant compte de la culture, de la structure de gouvernance, des projets en cours et à venir et des technologies utilisées.
- **Adopter une stratégie de transition agile** tenant compte de la culture, des risques et des impacts sur l'organisation.
- **Planifier et faire le suivi** du projet de transition agile qui doit être traité comme un projet organisationnel.
- **Choisir les méthodes agiles** qui conviennent aux besoins et au contexte organisationnel.
- Adapter les processus ou **définir un cadre méthodologique** dans lequel les méthodes agiles choisies sont mises en œuvre avec des principes directeurs.
- **Se doter d'outils** adéquats pour opérationnaliser les processus et donner de la visibilité à tous.
- **Former et accompagner** les gestionnaires, les membres d'équipes et les clients impliqués dans les projets de développement.
- **Évaluer l'application des pratiques agiles** à intervalles réguliers pour s'assurer que l'organisation obtiendra le rendement escompté et appliquer les adaptations requises pour y arriver.

Mise en garde : Il faut prendre garde de ne pas confondre les risques avec les problèmes rendus visibles. L'agilité est un processus d'inspection et d'amélioration continue, ce qui implique que son application mettra en évidence les problèmes d'une organisation. Cette dernière doit être prête à considérer les problèmes

soulevés comme des opportunités d'amélioration et c'est à elle qu'incombe de saisir ces opportunités pour atteindre ses objectifs. Autrement, l'agilité se verra accusée d'être la source des problèmes et son adoption sera remise en cause.

11.6.3 Stratégies de transition agile

Plusieurs stratégies de transition à l'agilité ont été recensées. Nous vous en proposons quelques-unes ci-après.

Par les gestionnaires

Dans une approche « par les gestionnaires », aussi appelée « par le haut », l'adoption de l'agilité provient d'un gestionnaire important qui sait communiquer et partager sa vision de l'agilité. Il est cependant difficile pour ce leader, même en donnant une direction claire à ses troupes, de traîner seul le fardeau d'un changement de cap aussi important. En effet, car il devra non seulement communiquer sa vision à un nombre parfois imposant de personnes, mais aussi éliminer les obstacles majeurs, générer des succès à court terme, mener et gérer plusieurs projets de changement et faire en sorte d'ancrer les nouvelles approches profondément dans la culture organisationnelle[1].

Par les équipes

L'approche « par les équipes », parfois appelée « par le bas », fait référence aux membres des équipes de développement logiciel qui sont à l'origine de la mise en œuvre des pratiques agiles. Typiquement, l'équipe commence à appliquer certaines des pratiques agiles des méthodes retenues, notamment celles qui s'appliquent pendant la réalisation. Ces équipes le font souvent à l'insu de la direction, du moins au début, car ils veulent d'abord se prouver à eux-mêmes que l'approche agile peut fonctionner dans leur contexte et leur culture avant de présenter une requête d'adoption officielle de l'agilité dans l'organisation. Le proverbe qui dit « *le pardon est plus facile à obtenir que la permission* » s'applique ici.

Après un premier succès en réalisation, les pratiques agiles peuvent s'étendre successivement ou en parallèle aux autres disciplines du développement logiciel : gestion des exigences, gestion de projet, architecture et déploiement. Puis, le succès aidant, on finit par appliquer l'agilité jusqu'au bureau de projet et aux équipes de gestion.

En sandwich

L'approche « en sandwich » est une combinaison des approches « par les gestionnaires » et « par les équipes », c'est-à-dire que l'agilité est souhaitée et amorcée par les équipes en plus d'être soutenue par la direction au niveau approprié. En général, elle est utilisée lorsqu'une ou plusieurs équipes ont eu du succès en appliquant l'agilité dans la phase de réalisation. Ensuite, elles utilisent ce succès

1. Adapté de John P. Kotter, *Leading change*, Harvard Business School Press, 1996.

comme levier lorsque la direction communique son intention d'étudier les méthodes agiles. Il y a alors synchronicité entre les désirs de la direction et ceux des équipes voulant étendre les pratiques au-delà de la réalisation.

Progressiste

L'approche « progressiste », aussi appelée l'approche « des petits pas » consiste à commencer avec un projet et à contaminer de plus en plus de projets. Elle est selon nous la plus populaire en raison de ses chances de succès dans des organisations de toutes tailles.

Typiquement, la transition commence avec une planification de l'adoption de l'agilité : raison d'être, vision, besoins, diagnostic de la préparation à l'agilité, stratégie d'adoption, rôles et responsabilités, affectations de personnel, sessions de formation, critères pour choisir les projets pilotes, etc. Ensuite, on utilise un petit nombre de projets pilotes, souvent un seul, dont on s'inspire pour définir et enrichir le cadre méthodologique des processus agiles. Ensuite, il est déployé à d'autres projets pilotes ou au reste des projets ciblés par l'organisation, comme illustré à la Figure 11.1.

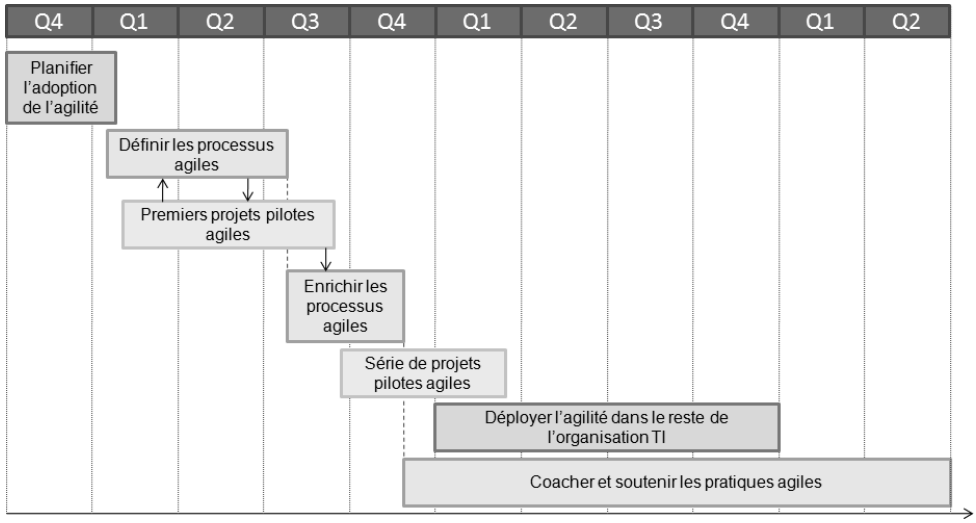


Figure 11.1 — Plan typique d'adoption progressiste de l'agilité à l'échelle organisationnelle[a].

a. Adapté de *Bridging the Gap Between Governance and Agility: Implementing Agile into your Organization,* par Emilio Di Zazzo et Douglas Page de CA Technologies, présenté à Agile Montréal en novembre 2010.

Afin d'assurer la pérennité des pratiques agiles, il est important de mettre en place une mécanique de coaching et de soutien. De cette manière, tout nouveau projet ou nouveau membre du personnel sera introduit adéquatement aux pratiques agiles adoptées de l'organisation.

Big Bang

L'approche *Big Bang* consiste à faire *tabula rasa* sur les méthodologies qui existent et tout démarrer ou redémarrer en mode agile. En général, les organisations qui adoptent le *Big Bang* sont dans les situations suivantes :

- Société en démarrage qui n'a pas de méthodologie existante.
- Société qui s'apprête à investir dans une série de projets risqués pour lesquels les méthodologies existantes se sont avérées inefficaces.
- Société qui a perdu le contrôle sur l'ensemble de projets dont aucun n'aboutit nulle part.

Une organisation qui applique l'approche *Big Bang* ne le fait pas aveuglément : il y aura nécessairement un sentiment d'urgence face au changement. Dans ces cas, plusieurs des membres de l'organisation ont été praticiens de l'agilité ailleurs et savent comment procéder, ou l'organisation est accompagnée d'experts en agilité tout au long de sa transition.

Cette approche n'est pas recommandée dans de grandes organisations où plusieurs projets ou équipes doivent démarrer en même temps car les experts (internes ou externes) qui accompagnent les équipes sont vite débordés. L'organisation risque alors de mal appliquer les pratiques agiles dans certains projets qui devront être redressés par la suite. Néanmoins, quelques grandes sociétés ont adopté cette approche et ont publié les résultats, notamment Yahoo et Salesforce.com.

11.7 ÉVALUATION PÉRIODIQUE DE L'APPLICATION DES PRATIQUES AGILES

Retour d'expérience : L'agilité inscrite au plan d'affaires

Au printemps 2004, je suis intervenue auprès d'une société qui développait des logiciels dans le domaine de la santé pour y faire un diagnostic de leurs processus au regard des meilleures pratiques contenues dans le CMMI. Le directeur du produit, avait écrit dans son plan d'affaires qu'ils utilisaient *eXtreme Programming* (XP) comme méthode de développement. Il faut se rappeler que les méthodes agiles n'étaient pas répandues à cette époque.

Arrivé sur place, le dirigeant me confirme que leur méthode de développement est bel et bien XP. J'ai donc commencé à chercher des traces laissées par l'exécution de la méthode XP en rencontrant les membres de l'équipe de développement. J'ai aussi rencontré la personne qui occupe les rôles de chef de projet et d'architecte principal et qui me confirme qu'ils appliquent XP.

Après avoir examiné chacune des pratiques d'XP, je constate qu'ils n'en appliquaient AUCUNE : pas de binômage, pas de rythme soutenable, pas de remodelage de code, pas de gestion du code source, pas de co-localisation des membres d'équipe. Mais le terme XP sonnait d'avant-garde dans leurs documents. Malheureusement, j'ai rencontré plusieurs cas dans l'industrie où la direction était persuadée que les bonnes pratiques étaient appliquées et s'étonnait du piètre résultat obtenu.

Sylvie

Ce n'est pas parce qu'une équipe, une division ou une organisation prétend appliquer l'agilité qu'elle le fait réellement ou efficacement. L'agilité mal appliquée n'apporte pas les bénéfices anticipés, et les risques de ne pas atteindre les objectifs fixés peuvent alors être élevés. Pour réduire ce risque, nous proposons aux organisations d'effectuer régulièrement un diagnostic des pratiques agiles. Les recommandations et les plans d'actions qui en découlent visent à corriger les mauvaises applications des pratiques et ainsi tendre plus efficacement vers les objectifs attendus de l'agilité. Pour être impartial et profiter d'un second regard, ces évaluations doivent préférablement être menées par des experts externes et/ou indépendants de la hiérarchie des groupes appliquant les pratiques diagnostiquées.

11.7.1 Le PATH : un outil de diagnostic de l'agilité

Nos collègues de Pyxis Technologies ont créé le PATH, un outil permettant de mesurer le niveau de maturité agile des organisations[1]. Ce modèle propose de mesurer le niveau d'agilité sous quatre axes différents, soit celui des **p**rocessus, des **a**ffaires, de la **t**echnologie et de l'aspect **h**umain[2]. En plus des quatre axes, le PATH introduit trois niveaux d'influence qui, appliqués aux quatre axes, constituent douze zones d'intervention (voir Figure 11.2) :

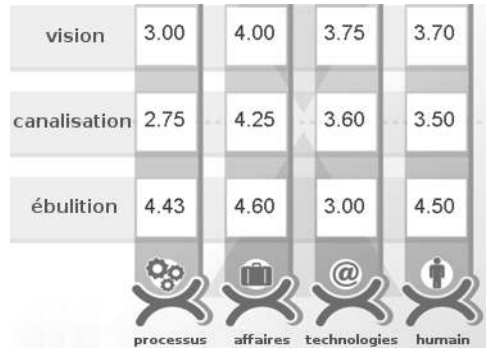


Figure 11.2 — Exemple de résultats d'un diagnostic du niveau de maturité agile d'une organisation avec l'outil PATH.

- **Vision** : ce niveau indique l'orientation à prendre dans le but d'atteindre les objectifs fixés. Ce niveau est fortement lié à la vision stratégique de l'entreprise.
- **Canalisation/dissémination** : ce niveau a comme principale préoccupation de mettre en place des mécanismes favorisant la collaboration, l'application de bonne pratiques et l'amélioration continue.
- **Ébullition** : le niveau 'ébullition' est celui des équipes de projet qui réalisent les solutions logicielles.

1. Pour plus d'information, vous pouvez vous référer au site suivant : http://pyxis-tech.com/fr/notre-offre/coaching/path .
2. Le nom « PATH » est issu de la première lettre de ces quatre axes

Pour chacune des zones, la cote de maturité des pratiques mises en œuvre est comparée avec celle de la vision organisationnelle. Cela permet de vérifier si l'application et la vision sont bien alignées et que les moyens mis en œuvre pour assurer cet alignement sont efficaces.

Avec les résultats de la figure 11.2, constatons comment un outil comme le PATH peut aider à mesurer le niveau de maturité agile d'une organisation :

- Axe des **Processus** : Les résultats de l'exemple illustrent le cas d'une organisation où les équipes (niveau d'intervention *ébullition*) comprennent mieux les processus associés à l'agilité que les dirigeants de l'organisation (niveau d'intervention *vision*). Cela peut expliquer pourquoi les moyens mis en œuvre (niveau d'intervention *canalisation*) pour soutenir les équipes dans leurs démarches sont insuffisants.
- Axe des **Affaires** : le diagnostic démontre également un bon alignement entre la vision d'affaires, les moyens déployés et la mise en œuvre par les équipes de projet.
- Axe des **Technologies** : la figure illustre que bien que les dirigeants aient une forte attente envers l'excellence technique des équipes. Ces dernières n'appliquent pas encore les pratiques reconnues par les praticiens agiles (ex. le développement piloté par les tests ou TDD).
- Axe **Humain** : semblable à l'axe des processus, les équipes appliquent des pratiques tenant compte des aspects humains sans toutefois être pleinement soutenues par une vision claire ni des moyens suffisants. Dans ce cas, il est à prévisible que ces pratiques ne seront pas pérennes.

En résumé

Une transition agile n'est pas un projet TI mais bien un changement organisationnel. Adopter l'agilité aura des impacts importants sur la façon de travailler des équipes et affectera aussi les gestionnaires et les dirigeants qui doivent apporter quelques modifications dans leurs façons de travailler.

Afin de bien pouvoir gérer l'adoption de l'agilité, ces gestionnaires doivent mettre leurs connaissances à niveau pour comprendre l'impact d'une telle adoption. Ils doivent s'attendre à plus de transparence dans le processus et de visibilité sur les problèmes.

Certaines pratiques agiles passent difficilement dans des cultures spécifiques et des solutions pratiques peuvent être apportées comme compromis. Il faut cependant prendre garde à apporter des modifications aux pratiques agiles pour les mauvaises raisons.

Quand la transition agile s'opère avec succès, l'organisation augmente son avantage compétitif. Le soutien des gestionnaires est cependant indispensable au succès d'une telle initiative.

12

Évolution des processus

Objectif

Dans ce chapitre, vous découvrirez qu'un processus organisationnel n'est pas quelque chose d'immuable. Vous comprendrez pourquoi il se doit d'être évolutif pour cadrer avec l'expérience terrain de votre organisation. Vous deviendrez critique face à sa qualité, ce qui vous amènera à le réévaluer sous différents aspects :

– Est-il maintenu et maintenable ?

– Est-il représentatif de ce qui se fait dans l'organisation ?

– Est-il diffusé et connu ?

– Est-il utile et adapté aux besoins de ses praticiens ?

Mise en situation : Philippe et les processus agiles

Philippe est cadre supérieur des TI dans une institution financière. Il est responsable de l'application des processus logiciels de son organisation. L'ensemble de ces processus représente plus de 1500 pages de descriptions, de guides, de gabarits, de procédures et de formulaires. Des sommes considérables d'effort et de coûts ont été investies dans ces processus logiciels depuis plus de quinze ans.

Ces processus décrivent notamment le cycle de vie en cascades des projets : parfois, ils débutent avec une étude d'opportunité pour découvrir les besoins d'affaires et sonder si une solution logicielle est envisageable. Invariablement, chaque projet a sa phase d'analyse préliminaire visant à déterminer une ébauche de la solution logicielle et l'enveloppe budgétaire requise. Cette phase est d'une durée moyenne de quatre mois et permet de ficeler les besoins d'affaires avec la solution proposée.

Si le projet est accepté par les utilisateurs payeurs, alors démarre la phase d'architecture détaillée comportant trois sections :

– l'architecture fonctionnelle avec sa liste de fonctionnalités décrites sommairement et un modèle des données d'affaires ;

– l'architecture organique avec les interfaces inter-systèmes et la structure de l'application ;

– l'architecture technologique avec la vue de déploiement et les orientations technologiques telles que le choix du langage de programmation et les plateformes de développement, de tests et de production.

Suite au contenu du document d'architecture détaillée, rédigé en moyenne en six à dix mois, une phase de planification est amorcée afin de présenter l'enveloppe budgétaire finale ainsi qu'un calendrier d'exécution détaillé, sous la forme d'un graphique de Gantt, auquel il ne manque que la date de début pour tout orchestrer dans ses moindres détails. Cette planification prend plus d'un mois en moyenne. Vient ensuite la phase pendant laquelle les analystes produisent un dossier par fonctionnalité décrivant en détail le comportement, les données utilisées, ainsi que de l'information de conception. Chaque dossier fonctionnel contient de 30 à 250 pages.

Philippe a récemment approuvé le démarrage d'un projet en mode agile et l'équipe se demande si elle doit néanmoins continuer d'appliquer des portions importantes du processus et de ses acquis. Philippe a donc consulté un expert en agilité pour y voir plus clair. Il a recommandé d'appliquer la majorité des livrables déjà présents au processus afin de ne pas trop dépayser les membres de l'équipe agile. Toutefois, il leur a suggéré de :

– revoir la pertinence du contenu et du contenant de chaque document, pour les épurer des sections ne convenant pas au contexte du projet ;

– proposer des améliorations au processus de rédaction et de validation pour augmenter l'efficacité de la production des livrables ;

– démarrer le développement de la solution logicielle plus tôt et continuer la rédaction des documents pendant le projet.

12.1 DOCUMENTER SON PROCESSUS

La documentation des processus agiles dans les organisations n'est pas un sujet unanime. Peut-être parce que :

- Les approches agiles sont des philosophies de travail concernant plus le savoir-être des individus que leur savoir-faire.
- Les approches agiles sont peu prescriptives, et que la documentation de quelques livrables, rôles et pratiques ne requiert pas de références particulières.
- Les processus organisationnels sont parfois considérés comme immuables alors que les approches agiles proposent qu'ils soient soumis à l'amélioration continue.

Bien que ce soit des arguments valables, la documentation du processus organisationnel peut constituer une condition de succès à l'adoption de l'agilité. En particulier au sein des grandes organisations, où la seule communication directe avec

tout le personnel n'est pas réaliste pour assurer une compréhension commune et une application uniforme du processus. Les raisons utiles de documenter le processus organisationnel sont les suivantes :

- Traduire les objectifs stratégiques sous la forme d'une méthodologie. L'adoption de l'agilité doit être basée sur un plan stratégique entre les affaires et les TI[1]. La documentation du processus reflète comment l'organisation a décidé de mettre en œuvre les approches agiles.
- Soutenir la prise de décisions en définissant les règles et les principes retenus par l'organisation.
- Communiquer aux membres de l'organisation les attentes autour des rôles, des responsabilités, des livrables et des pratiques prévus par le processus.
- Suivre et évaluer le processus, afin de vérifier que l'organisation fait ce qu'elle dit et améliore continuellement son processus de manière à ce qu'il soit adapté à sa réalité et ses besoins changeants.

La conformité à des normes de qualité n'est pas à elle seule une raison valable de documenter le processus d'une organisation et il importe d'identifier les objectifs et les besoins des individus qui y font référence[2].

12.1.1 Portée de la documentation

Plusieurs organisations ont documenté leur processus de développement, et le degré de complexité varie selon leur contexte. Nous avons pu constater combien le spectre de la documentation est nuancé d'une organisation à une autre.

Dans sa forme la plus légère, la documentation du processus tient sur quelques pages identifiant les rôles, les livrables et les activités prévues. Une telle documentation se veut avant tout un aide-mémoire et un support à la discussion lorsque les concepts généraux et les grandes étapes du processus ont besoin d'être rappelés ou diffusés.

Dans sa forme la plus complète, la documentation décrit en détail les processus de toute l'organisation, du conseil d'administration aux membres des équipes projets. Elle peut notamment décrire toutes les responsabilités des rôles sous la forme d'un RACI[3], fournir les gabarits de tous les livrables et détailler le processus qualité de chaque pratique. Une telle documentation tiendra sur plusieurs centaines de pages.

1. Se référer au chapitre *11 Gouvernance*.
2. La production d'une documentation qui est utile, utilisable et utilisée pour ses destinataires est traitée au chapitre *14 Agilité et documentation*.
3. RACI: Matrice des rôles et responsabilités, par livrable des processus, dans laquelle sont indiqués le « Responsable », les « Approbateurs », ceux qui sont « Consultés » et ceux qui sont tenus « Informés » sur le contenu. Se référer au chapitre *11 Gouvernance* pour plus de détails.

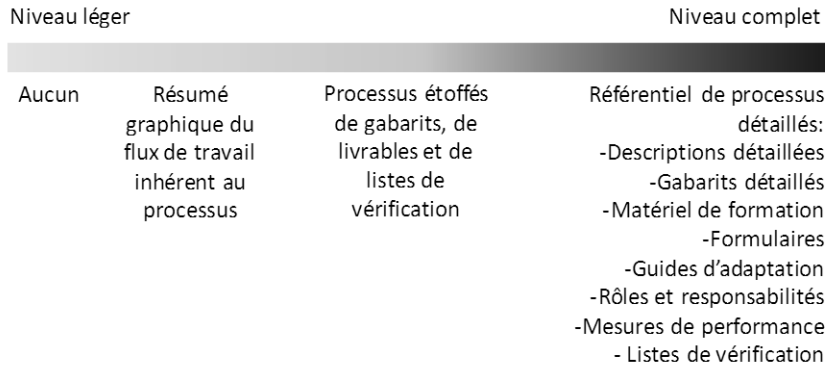


Figure 12.1 — Le degré de documentation est variable d'une organisation à une autre[a].

a. Les listes de vérification, en anglais « *checklist* », permet de vérifier méthodiquement les caractéristiques attendues d'une pratique ou d'un livrable. Les procédures AQ, décrites plus loin dans ce chapitre, et la définition de terminé sont des exemples de telles listes.

Le degré de complexité de la littérature agile varie également d'une méthode à une autre. Les méthodes adoptées par une organisation peuvent donc avoir un impact sur la portée de la documentation de son processus.

Malgré tout, une transition vers l'agilité viendra immanquablement remettre en cause le contenu et la forme de la documentation d'une organisation. Ces dernières peuvent alors réagir de trois manières différentes :

- Ne rien remettre en cause : pour des raisons de culture, de conformité, ou simplement parce qu'elles ne connaissent pas de meilleures façons de documenter le processus, certaines organisations ignorent l'opportunité de revoir leur documentation détaillée et uniforme et d'y trouver des formes de gaspillage coûteuses en temps et en argent.
- Ne plus exiger de documentation et laisser les équipes gérer cet aspect : cette réaction est souvent exagérée car elle rejette du même coup toute la richesse de la documentation que l'organisation a pu réaliser depuis des années.
- Adapter graduellement le processus à l'agilité : selon des principes d'amélioration continue, cela permet d'évaluer les éléments de la documentation et d'épurer graduellement les formes de gaspillage autour de son usage, de son entretien et de sa consultation. C'est l'approche que nous préconisons.

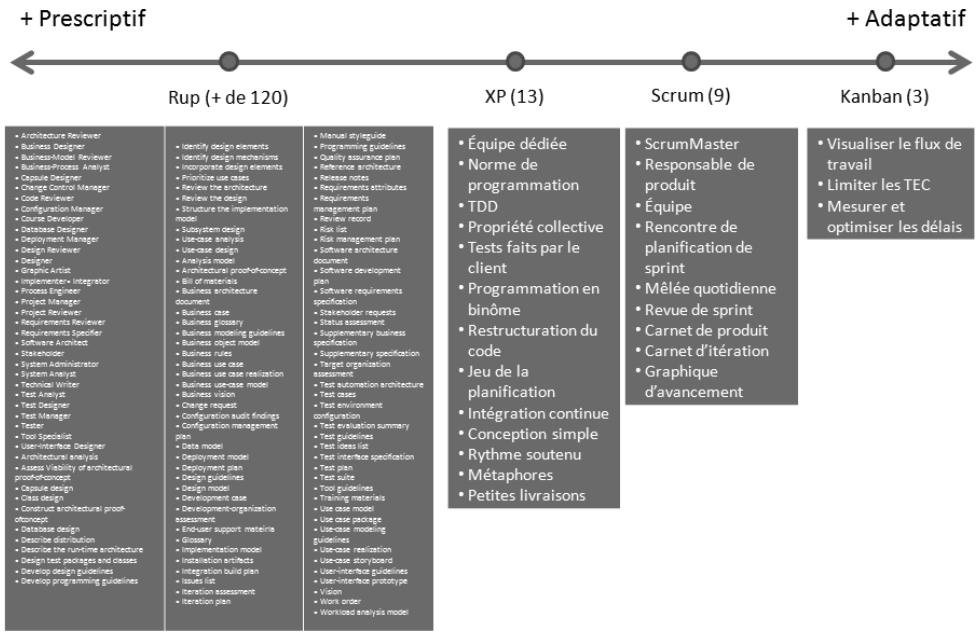


Figure 12.2 — Complexité basée sur le nombre de rôles, pratiques et livrables définis par méthode agile[a].

a. Tiré de l'ouvrage *Kanban and Scrum - making the most of both*, des auteurs Henrik Kniberg et Mattias Skarin, http://www.infoq.com/minibooks/kanban-scrum-minibook, consulté en décembre 2009. Nous sommes conscients que la liste des pratiques et livrables de RUP est illisible mais nous ne tenions qu'à montrer le volume important qu'ils représentent.

12.1.2 Utiliser la documentation existante

Au moment d'entamer une transition agile, une organisation sans processus documenté peut vouloir introduire une référence afin de diffuser la manière dont l'agilité est envisagée. Dans ce contexte, une documentation visuelle, tenant sur moins de cinq pages, décrivant le flux de travail, identifiant les interactions et les points de synchronisation entre les différents rôles et équipes peut être largement suffisante. L'organisation sera en mesure d'enrichir cette base au fur et à mesure que ses besoins émergeront.

Dans le cas d'une organisation ayant déjà une documentation de ses processus, il est préférable de ne pas en faire table rase mais plutôt de s'adapter graduellement aux approches agiles. Cette technique réduit l'introduction de nouveaux termes et l'inconfort causé par l'abolition complète d'un processus connu et maîtrisé.

Prenons par exemple les livrables. Mis à part l'introduction de quelques nouveaux livrables comme le carnet de produit et de la définition de terminé, la plupart des livrables n'ont pas nécessairement besoin d'être modifiés au départ de la transition agile. C'est le cas notamment des livrables clés de l'organisation, comme les dossiers d'architecture et les dictionnaires de données qui peuvent continuer d'être utilisés

dans leur forme actuelle et être bonifiés par les activités d'amélioration continue. L'adaptation graduelle du processus implique cependant qu'il existera des niveaux intermédiaires des livrables. Mises à part quelques considérations d'uniformité, cela devrait avoir des conséquences modérées puisque les éléments essentiels et pertinents devraient être communs entre les versions courantes et actuelles des livrables.

Outre l'introduction de quelques nouveaux rôles, comme le Scrum Master ou le responsable de produit, la description des postes ne sera pas *a priori* très différente suite à la transformation du processus. Les développeurs resteront des développeurs, et ce sera la même chose pour les analystes, les architectes, les gestionnaires et la plupart des autres corps de métier de l'organisation. Cependant, les modes de communication, les interventions, le style de leadership et la collaboration seront transformés. Ce sont donc les activités du processus qui seront les plus impactées. C'est également le cas de la matrice RACI. Les approches agiles préconisent la responsabilisation et l'auto-organisation, ce qui a pour effet de déplacer l'imputabilité de l'approbation des livrables des responsables hiérarchique vers les consommateurs. Au chapitre *11 Gouvernance*, nous avons expliqué comment une approche agile avait déplacé le rôle de l'approbateur des dossiers de conception de l'architecte, expert du métier, vers l'équipe de projet, consommatrice de l'information. En effet, le consommateur utilisant l'information est souvent plus en mesure de juger de son utilité et de sa qualité qu'un responsable hiérarchique qui consulte le document pour seule fin de validation.

Nous conseillons donc de commencer la révision de votre processus organisationnel par les activités, les nouveaux rôles et les livrables introduits et de revoir le reste du processus par le biais des activités d'amélioration continue.

12.1.3 Recommandations sur la manière de documenter

Les approches agiles reposent beaucoup sur la capacité de jugement des individus. C'est pourquoi, malgré quelques recommandations, la littérature agile n'exige pas de règles strictes. Par exemple, certains praticiens recommandent :

- Que les tâches d'un carnet d'itération n'excèdent pas deux jours de travail. L'intention est d'augmenter la visibilité en divisant le contenu de l'itération en petits morceaux faciles à gérer, en partageant le travail entre les individus et en suivant la progression de l'itération. La recommandation ici est d'avoir les tâches les plus petites possibles et deux jours semblent un temps raisonnable pour accomplir quelque chose.
- Que tous les participants à la mêlée quotidienne répondent aux questions suivantes : *Qu'ai-je fait hier ? Que ferai-je aujourd'hui ? Et quels obstacles me bloquent ou me ralentissent ?* Ces questions sont un modèle et l'intention est de susciter la communication ainsi qu'aider l'équipe à prendre conscience de la santé de l'itération, bien plus que de répondre aux trois questions.

Les précédents exemples démontrent que le plus important est l'intention d'une pratique plutôt que le suivi d'une procédure stricte. Nous vous recommandons d'être attentifs lorsque vous documentez votre processus agile, en écrivant les intentions

d'une pratique et les résultats attendus d'un livrable, puis de laisser les individus décider comment ils les réaliseront.

Conseil : Prenez garde à ne pas transformer une recommandation, qui donne un indice sur la portée d'une pratique et l'ampleur d'un livrable, en une règle absolue qui réduit l'initiative et l'autonomie des individus.

Portez une attention particulière au vocabulaire introduit par les approches agiles. Les approches agiles ne sont encore largement documentées qu'en anglais et malheureusement, la francisation des termes amène souvent un sens différent. Dans la mesure du possible, éviter de nommer les objets du processus sous différentes appellations ou l'utilisation de deux langues. Cela facilitera les échanges et la compréhension de ceux appliquant le processus. Un exemple que nous rencontrons souvent, c'est l'utilisation de l'acronyme *PO*, en anglais *Product Owner*, pour désigner le responsable de produit. Le choix de garder le terme anglais ou l'utilisation de l'acronyme RP est moins ambigu.

12.2 GESTION DU PROCESSUS

L'objectif de la gestion du processus est de réduire les coûts d'opération, les délais de traitement et d'augmenter la qualité des résultats. Cette gestion peut se faire selon deux axes : l'amélioration continue et l'assurance qualité.

12.2.1 Amélioration continue

L'amélioration continue fait évoluer le processus de manière à s'assurer qu'il est toujours adapté et optimisé autour de la réalité de l'organisation. Nous vous proposons dans cette section un modèle inspiré de la roue de la qualité de Deming[1] (Figure 12.3).

La roue de Deming est une méthode de gestion de la qualité visant à introduire un cycle d'amélioration continue en quatre étapes :

- **Planifier** : étape d'introspection visant à évaluer l'application du processus organisationnel. *S'il est mal appliqué, est-ce parce qu'il est mal interprété ? Qu'il n'est pas reconnu ? Qu'il est mal adapté ?* Une fois le problème identifié, les causes fondamentales sont recherchées dans le but de proposer des alternatives. Identifier la racine du problème est important, car un problème réglé en surface resurgira sous une nouvelle forme. Ensuite, une proposition est choisie en tentative de résolution du problème.
- **Développer** : à cette étape, la proposition identifiée précédemment est mise en œuvre à titre d'essai.
- **Contrôler** : permet de mesurer les résultats de la mise en œuvre. La proposition donne-t-elle les résultats suffisants et attendus ?

1. Wikipedia, http://fr.wikipedia.org/wiki/Roue_de_Deming, consulté en décembre 2010.

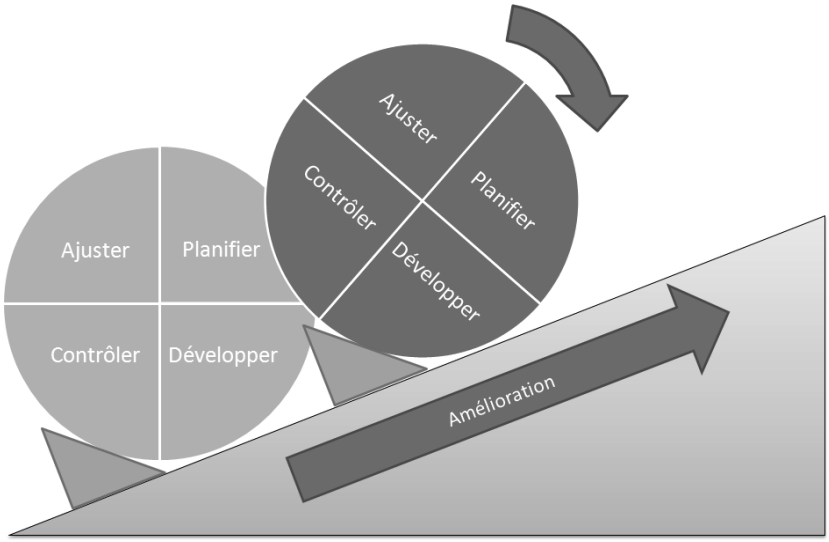


Figure 12.3 — La roue de Deming.

- **Ajuster** : si la proposition donne les résultats attendus, elle devient la solution à mettre en œuvre. Autrement, le cycle est repris jusqu'à ce qu'une proposition adéquate soit mise en œuvre.

La mise en situation suivante explique comment la roue de Deming peut être appliquée dans un contexte d'amélioration continue sur processus organisationnel agile.

Mise en situation : Amélioration de la pratique de la mêlée

Lors d'une rétrospective, une équipe décide de se questionner à propos de l'efficacité de ses mêlées quotidiennes.

Planifier

Le déroulement prévoit que chaque membre de l'équipe diffuse aux autres membres les travaux qu'il a effectués depuis la dernière rencontre en répondant aux trois questions classiques, telles qu'énoncées au point « planifier » précédemment.

Pendant la rétrospective, l'équipe détermine que ces questions incitent souvent les participants à se rapporter au Scrum Master plutôt que de se synchroniser entre eux. Après réflexion, l'équipe identifie une alternative pour transformer la dynamique de la rencontre quotidienne en proposant de nouvelles questions. Plutôt que de se questionner sur le travail accompli par les membres de l'équipe, l'inspection pourrait se concentrer sur l'avancement des scénarios utilisateurs. Pour chaque scénario utilisateur, l'équipe répondra maintenant aux questions suivantes :

– Qu'est-ce qui a été fait sur ce scénario hier ? Qu'est-ce que le produit logiciel fait de plus aujourd'hui ?

– Que reste-t-il à faire sur ce scénario pour le compléter ? À quelle date prévoyons-nous de le terminer ?

– Quels sont les obstacles nous empêchant de compléter le scénario ?

Ensuite, l'équipe consultera son carnet d'itération pour se questionner :

– Notre engagement est-il toujours réaliste ? Sinon, quel est notre plan d'action et qui devons-nous prévenir ?

Développer

L'équipe décide de mettre en application leur nouveau modèle dès le lendemain et de réévaluer la solution à la prochaine rétrospective.

Contrôler

Lors de la rétrospective suivante, l'équipe confirme que ce nouveau modèle aide à mieux atteindre les objectifs fixés par la tenue des mêlées quotidiennes.

Ajuster

L'équipe prévoit faire un retour d'expérience afin de proposer cette amélioration au reste de l'organisation et la documenter dans le processus organisationnel.

La précédente mise en situation démontre comment les acteurs du processus peuvent être impliqués directement dans son amélioration. D'ailleurs, l'organisation a intérêt à consulter les résultats des activités d'amélioration continue existantes au sein des équipes, comme les mêlées quotidiennes ou les rétrospectives, pour diffuser et appliquer les meilleures idées issues des expériences du terrain.

12.2.2 Assurance qualité

Pour s'assurer que son processus documenté est utile et utilisé, une organisation peut se doter d'une politique d'assurance qualité (AQ) visant à définir les critères sur lesquels l'application du processus sera évaluée. Documenter le processus d'AQ permet de se référer à ce qui est prévu et attendu du processus et permet de discerner une mauvaise application d'un besoin d'amélioration.

Il faut cependant prendre garde à ce que le processus d'AQ ne soit pas un vœu pieu : si une organisation le définit, elle devrait également l'appliquer. L'AQ ne devrait pas non plus devenir un outil de contrôle au service des gestionnaires, en particulier dans un contexte agile. L'intention devrait plutôt être de responsabiliser tous les individus à l'importance de la qualité.

La documentation de l'AQ est un outil permettant aux individus de vérifier eux-mêmes l'utilité et la pertinence des activités auxquels ils participent et des livrables qu'ils consultent ou produisent. Pour assurer l'adhésion des individus, l'AQ devrait donc être connue, diffusée, réaliste et ajustée à la maturité de l'organisation. Cette adhésion est importante, car l'application de l'AQ est quelque chose de sérieux, qui demande beaucoup de rigueur.

Voici quelques critères d'AQ que nous recommandons d'appliquer au processus organisationnel :

- L'AQ est l'affaire de tous, et n'importe quel intervenant est en mesure de rapporter et corriger une non-conformité.
- Tous les livrables et toutes les pratiques doivent être conformes aux objectifs définis.

- La responsabilité de validation d'un livrable ou d'une pratique devrait être attribuée à la personne la plus proche possible de sa réalisation. Elle doit également être capable d'objectivité, c'est-à-dire avoir un minimum de recul pour pouvoir évaluer objectivement les critères de qualité.
- N'importe quel membre de l'organisation devrait être en mesure d'adresser une requête au responsable de l'AQ d'un livrable ou d'une pratique, afin de :
  - se faire conseiller sur le livrable ou la pratique ;
  - soumettre une proposition d'amélioration ;
  - demander d'appliquer les activités d'AQ dans un projet spécifique, pour un livrable ou une pratique donné ;
- Un mécanisme de non-conformité doit être appliqué afin d'assurer que les non-conformités soient résolues dans les délais attendus ;
- Un mécanisme de remontée des non-conformités non résolues dans les délais attendus doit être appliqué en dernier recours, lorsque tous les autres mécanismes de traitement de non-conformité se sont avérés inefficaces. Toutefois c'est notre souhait de ne jamais les utiliser. Le cas échéant, la remontée devrait se faire vers la personne ou le groupe responsable de l'approbation.

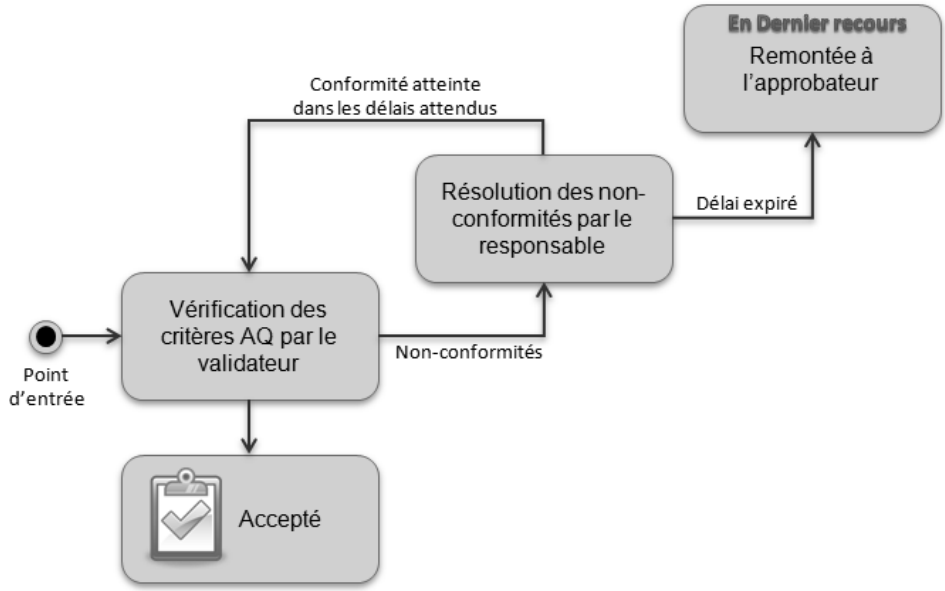


Figure 12.4 — Exemple de processus d'assurance qualité.

L'engagement envers la qualité est l'un des avantages de l'agilité[1]. Et pourtant, plusieurs croient à tort que l'AQ n'est pas compatible avec les principes des approches agiles. Cela dépend énormément de la manière dont l'AQ est définie et appliquée.

1. Cet aspect est traité au chapitre *1 Qu'est-ce que l'agilité ?*

Voici quelques conseils assurant la compatibilité entre l'AQ et les principes agiles :

- l'organisation doit être assurée que les individus sont en mesure de prendre en charge l'AQ ;
- l'AQ doit porter sur les livrables et les pratiques, et non pas sur les individus[1] ;
- Lorsqu'une non-conformité est détectée et corrigée, il est préférable de la soumettre au processus d'amélioration continue pour éviter qu'elle ne se reproduise ;
- la discussion verbale doit être préconisée, dans la mesure du possible, afin de réduire les risques de susceptibilité et de mauvaise interprétation ;
- la remontée doit toujours rester un ultime recours que personne n'a envie d'appliquer.

Le Tableau 12.1 donne un exemple de documentation de l'AQ d'un livrable agile.

Tableau 12.1 — Processus AQ du livrable *Incréments de logiciel*.

1. Validateur	Le responsable de produit
2. Fréquence	À chaque itération
3. Critères	Les nouvelles fonctionnalités répondent aux conditions de succès des scénarios correspondants ; L'incrément de logiciel est conforme à la définition de terminé ; Seuls les scénarios complètement terminés composent l'incrément de logiciel ; L'incrément de logiciel est déployé sur un environnement à la disposition du responsable de produit.
4. Traitement attendu de la non-conformité	L'équipe de projet voit à résoudre les non-conformités identifiées avant de s'engager sur une nouvelle itération ; Les fonctionnalités non conformes et impossibles à corriger dans des délais raisonnables sont retirées de l'incrément de logiciel.
5. Délai de résolution	Avant la planification de la prochaine itération, au maximum 2 jours.
6. Escalade	Refus de la livraison de l'incrément de logiciel par le responsable de produit, basé sur le principe que seul un incrément de logiciel fonctionnel de qualité peut servir de mesure de progression d'un projet.

Le précédent exemple peut sembler sévère. Cependant, il assure que l'engagement que l'équipe prend envers le responsable de produit est sérieux. Ce processus n'impose pas à l'équipe d'avoir terminé toutes les fonctionnalités de l'engagement, mais il oblige cependant à ce que celles dites *terminées* le soient vraiment et que l'incrément de logiciel puisse être déployé en production avec la qualité attendue. Dans le cas contraire, le responsable de produit a l'autorité de refuser l'incrément, parce qu'entre le responsable de produit et l'équipe de projet, il y a :

1. Les rôles et responsabilités sont plutôt des aspects appartenant à la gestion des individus.

- une entente sur les conditions de succès avant l'itération ;
- un contrat de qualité sous la forme d'une définition de terminé ;
- un engagement sur les fonctionnalités à livrer à la fin de l'itération ;

Ces règles de l'AQ sont raisonnables et devraient être acceptées et reconnues par l'équipe et le responsable de produit.

Ci-après un autre exemple, portant cette fois sur une pratique.

Tableau 12.2 — Processus AQ de la pratique de la *rétrospective d'itération*.

1. Validateur	Responsable des pratiques agiles
2. Fréquence	Une à trois fois par projet
3. Critères	Le plan d'action doit répondre à la règle **SMART** (Spécifique, Mesurable, Atteignable, Réaliste, limité dans le Temps) ; Les actions doivent être pérennes et concrètes (ex. ajouter une nouvelle colonne dans le tableau des tâches, insérer un nouvel item dans le carnet de produit, soumettre les modifications du processus au responsable des pratiques agiles, modifier la définition de terminé, modifier les normes de programmation, etc.) ;
4. Traitement attendu de la non-conformité	Accompagner le Scrum Master pour qu'il puisse à son tour amener son équipe à définir des actions SMART et pérennes ainsi qu'identifier les points d'amélioration les plus critiques et importants.
5. Délai de résolution	Avant la prochaine rétrospective (une itération). Le responsable des pratiques agiles assiste à la prochaine rétrospective et accompagne le Scrum Master une seconde fois si nécessaire.
6. Escalade	Faire une réunion avec le Directeur de projet, le Scrum Master concerné et le responsable des équipes agiles sur les causes de la récurrence des rétrospectives non-conformes et le besoin d'un programme d'accompagnement et de formation.

L'amélioration continue est un concept important de l'agilité et une organisation peut vouloir s'assurer qu'elle est correctement appliquée par les équipes de projets. L'intention n'est pas de prendre les équipes en défaut, mais plutôt de s'assurer que le résultat de leurs rencontres atteint les objectifs visés. C'est pourquoi la règle de l'AQ ne détermine pas comment doit se dérouler la rencontre, mais vérifie plutôt le résultat obtenu. En cas de non-conformité, la règle de l'AQ prévoit d'accompagner le Scrum Master qui a peut-être besoin de parfaire ses compétences en matière d'animation pour être en mesure de rendre service à son équipe.

Les deux précédents exemples décrivent comment tous les individus peuvent participer à l'AQ du processus organisationnel, tout en étant responsables des résultats et avec des critères respectant les valeurs et principes de l'agilité.

Financement de l'assurance qualité

D'ordinaire, c'est l'organisation qui finance les activités de l'AQ, parce que ce sont les gestionnaires qui y trouvent de l'intérêt. Mais, il peut être intéressant de financer les activités d'audit de l'AQ par les projets. Évidemment, cela demande une certaine

maturité car une organisation veut prévenir qu'une équipe cherche à se soustraire aux processus de l'AQ. Cependant, lorsque l'équipe finance son propre audit, cela donne beaucoup de crédibilité aux recommandations et démontre combien l'AQ est une valeur ajoutée du projet.

12.2.3 Diffusion du processus

Chaque modification au processus organisationnel doit d'être diffusée pour que les individus soient en mesure d'être informés du changement. Nous recommandons :

- que l'information soit accessible par tous, par exemple via un wiki, un intranet ou une application spécialisée en gestion de contenu documentaire ;
- qu'un mécanisme de diffusion soit instauré afin de communiquer les changements apportés sur le processus organisationnel[1].

En résumé

La documentation du processus organisationnel permet de définir les résultats et les objectifs attendus par son application. La transition à l'agilité donne l'opportunité aux organisations de revoir leur processus organisationnel. Pour certaines, cela se traduit par l'introduction d'une documentation de référence, pour d'autres, cela demande une révision de celle en place.

Le principe d'introspection et d'amélioration continue des approches agiles s'applique également à l'entretien du processus organisationnel. Cela permet à une organisation d'atténuer le risque d'une transition en ne faisant pas table rase des bienfaits de la documentation actuelle.

Si un processus est documenté, il peut être attendu qu'il le soit correctement. Une politique d'assurance qualité permet d'évaluer la bonne utilisation du processus organisationnel et de vérifier qu'il apporte les résultats attendus. Autrement, il devrait être soumis au processus d'amélioration continue.

Bien qu'il faille porter une attention particulière à la façon de documenter le processus et à la politique d'assurance qualité, ces éléments peuvent être compatibles avec les valeurs et les principes agiles.

1. Le chapitre *13-Soutien au développement* décrit comment la diffusion et l'accessibilité de la documentation peut être faite de manière proactive, par les équipes de soutien au développement.

13

Soutien au développement

Objectif

Dans ce chapitre, vous découvrirez l'essentiel des pratiques recommandées pour soutenir l'adoption de l'agilité auprès des personnes impactées, car la seule volonté des membres des équipes ne suffit pas à réussir la transition agile. Vous verrez comment les mécanismes de soutien au développement peuvent être adaptés en fonction de la taille de la direction des TI – une dizaine, plusieurs dizaines, quelques centaines ou plusieurs milliers d'individus – et comment déterminer le budget approprié au soutien requis.

Mise en situation : Bernard et la transformation de l'informatique

La haute direction d'une grande société d'assurance envisageait d'adopter l'agilité pour résoudre les problèmes d'insatisfaction des lignes d'affaires face aux solutions logicielles et résoudre ainsi les dépassements budgétaires. Avant de se lancer, ils demandèrent à Bernard, leur vice-président des TI, de faire l'analyse d'impacts d'une telle transition.

Bernard comprend que toutes les sphères d'activités sous sa direction (près de 500 personnes) seront touchées par cette transition. Et c'est sans compter les lignes d'affaires et la gestion de l'infrastructure corporative qui seront aussi impactées. Après s'être documenté, Bernard s'est adjoint le soutien d'un expert agile pour l'aider à produire une analyse d'impact sur les processus en place. Après avoir décrit à sa direction les résultats de l'analyse d'impact et les retombées positives potentielles, Bernard recommande que l'adoption de l'agilité devienne un programme chapeautant un ensemble de projets agiles appuyé par une équipe responsable du « soutien au développement ». La mission de cette dernière consiste à accompagner les équipes de développement afin qu'elles adoptent les nouvelles pratiques agiles.

Compte tenu de la taille du groupe TI, Bernard choisit de commencer son programme par un seul projet pilote, car le contexte de dizaines de projets en cours avec la méthode traditionnelle ne se prête pas à une stratégie « big bang[1] ». De toute façon, il n'y a personne en place pour aider tout ce monde à changer leur façon de travailler. Il prévoit que ce programme durera entre trois et cinq ans, soit le temps estimé pour transformer l'ensemble des TI à l'agilité.

Bernard commence la mise sur pied de l'équipe de soutien au développement avec trois personnes. Même si elle ne développe pas de logiciels, l'équipe décide d'être un exemple de ce qu'elle promeut, en appliquant elle-même une démarche agile. Pour commencer, elle se donne une vision d'accompagnement, rédige une mission et définit les livrables qu'ils auront à produire :

– une charte de projet : vision, objectifs, rôles, leviers, risques, etc. ;

– un carnet de produit priorisé, dont Bernard est responsable ;

– un plan de livraison pour les six prochains mois ;

– l'analyse détaillée des items prioritaires pour couvrir les deux premières itérations ;

– un plan d'itération pour la première itération.

De cette façon, l'équipe s'est dotée d'un plan d'intervention adapté à la réalité de la transition grâce au processus d'amélioration continue auquel l'agilité les soumet.

13.1 LA MISSION DU SOUTIEN AU DÉVELOPPEMENT

La transition agile d'une organisation ayant appliqué une méthodologie traditionnelle depuis plusieurs années crée de l'incertitude dans les équipes de développement. Cette incertitude se manifeste sous la forme de questionnement de la part des individus :

- Que deviennent mon rôle et mon titre ?
- Qu'est-ce qui change dans le processus ?
- Quels outils vais-je utiliser ?
- Comment établir un carnet de produit ? Est-ce à moi de le faire ?
- Comment mener une rétrospective ?
- Les livrables seront-ils les mêmes ?
- Pourquoi le client fait-il maintenant partie de l'équipe ? Il ne comprend rien à l'informatique, alors comment ferons-nous pour nous comprendre ?
- Qu'est-ce que « Scrum » et « TDD » ?
- Dois-je vraiment écrire mes tests automatisés AVANT l'écriture du code ?
- Ah bon, ce n'est plus MON code mais c'est celui du client ?
- Comment utilise-t-on le nouvel outil de gestion du code ?

Cette liste de questions pourrait s'allonger sur plusieurs pages. Chacune de ces questions mérite de trouver une réponse afin d'éviter les pertes de temps inutiles et nuire au plein rendement de l'ensemble des membres des équipes.

1. Se référer au chapitre *11 Gouvernance*, pour les stratégies d'adoption.

Les individus les mieux outillés pour soutenir les équipes de développement sont ceux détenant le savoir, le savoir-faire et le savoir-être adéquat. Ils sont par conséquent les meilleurs candidats pour être responsables du « soutien au développement ». Par exemple, cela peut être un formateur ou un accompagnateur externe. Cela peut être un « champion » interne maîtrisant bien les concepts et les pratiques agiles et qui, surtout, arrive à très bien les communiquer. Un groupe de soutien au développement est d'abord et avant tout un centre de services et d'expertise dont chacun des membres doit apporter une valeur ajoutée à ceux qu'ils desservent.

13.1.1 Conséquences d'un manque de soutien au développement

Dans un contexte de transition agile, il est plus facile pour une équipe de moins de quinze membres de se convaincre, se soutenir et capitaliser sur les succès obtenus. Mais dans un plus grand groupe, de plusieurs dizaines de personnes, voire des centaines, les mécanismes de communication ne sont pas les mêmes. La diffusion d'une leçon apprise, d'une amélioration apportée, d'une nouvelle pratique ou de tout autre changement aux façons de faire doit non seulement se faire efficacement au sein des équipes, mais également au reste de l'organisation. Dans ces cas, l'affectation d'un ou plusieurs responsables du soutien au développement permet de s'assurer de la compréhension de tous face aux pratiques, aux valeurs et aux principes agiles. Sans ces responsables, l'organisation risque l'échec de l'agilité et l'abandon des pratiques, comme décrit dans la mise en situation ci-après.

Mise en situation : Alfred et la pseudo-agilité

Alfred est directeur du développement logiciel d'une grande société de distribution. Sa direction compte 120 employés répartis comme suit :

– 20 employés affectés à l'entretien des systèmes, soit la réalisation des demandes de changement ;

– 10 employés affectés au service à la clientèle et à l'infrastructure ;

– 90 employés affectés à huit projets de développement.

Comme pour plusieurs sociétés appliquant des méthodes traditionnelles, celle d'Alfred vit des dépassements budgétaires, des dépassements de dates de livraison, et de l'insatisfaction des lignes d'affaires face aux solutions développées.

Deux des équipes de projets décident d'essayer une méthode agile. Elles en parlent à Alfred qui leur donne son accord, tenté par les arguments de ses employés qui lui promettent des solutions utiles à moindres coûts. Prudent, Alfred exige néanmoins que l'ensemble des livrables de la méthodologie actuelle soient produits avant de lancer l'initiative. De plus, Alfred refuse que les utilisateurs ciblés viennent se joindre à l'équipe. Il n'ose pas le demander aux responsables des lignes d'affaires car il est convaincu qu'on le croira excentrique. À la place, il demande à ce qu'un analyste joue ce rôle. Et parce que les budgets de formation sont quasiment épuisés pour l'année, il n'envoie qu'une personne par équipe à une formation de deux jours sur la méthode agile choisie. Ces personnes deviennent responsables de former les autres personnes après leur formation.

Les projets commencent donc sans que personne ne sache vraiment comment établir un carnet de produit. L'analyste est incapable de répondre aux questions des développeurs sans consulter les clients, ce qui peut prendre des jours car ils sont tous très occupés. Les spécifications prennent plusieurs semaines à être décrites, ce qui devient un obstacle à la progression de l'équipe. Ne sachant pas comment travailler de manière incrémentale, l'équipe décide de consacrer ses deux premières itérations à l'architecture détaillée et la conception de fonctionnalités. Et même s'ils ne sont pas très ponctuels, tous les matins, les membres d'équipes font une courte rencontre de quinze minutes pour échanger sur les travaux en cours et ceux complétés.

Quand vient le temps de faire une rétrospective, seules les embûches mineures font surface et quelques améliorations sont apportées. Après la deuxième rétrospective, l'équipe se demande à quoi sert vraiment cette pratique car elle n'y voit pas de valeur ajoutée et décide de l'abandonner, au même titre que toutes les autres pratiques dont ils ne saisissent pas l'utilité :

– L'entretien du carnet de produit, parce qu'ils n'en ont pas et ne savent pas comment faire ;

– la planification d'itération, parce qu'ils n'ont pas de carnet défini ;

– les pratiques d'ingénierie agile, parce qu'ils ne les ont pas apprises ;

– les démonstrations, ou revues de sprint, parce qu'ils n'ont rien à montrer avant la cinquième itération, selon eux.

Après trois mois, ils décident d'abandonner l'agilité parce que ça ne fonctionne pas mieux que la méthode traditionnelle.

En réalité, les deux équipes d'Alfred n'ont JAMAIS fait d'agilité. Appliquons ici la technique des « pourquoi » successifs afin d'atteindre la cause fondamentale de cette mise en œuvre ratée.

1. Pourquoi ont-ils abandonné l'agilité ? Parce qu'ils ne l'ont jamais appliquée réellement ;

2. Pourquoi ? Parce qu'ils ont manqué de connaissances et de compétences ;

3. Pourquoi ? Parce que la formation a été insuffisante ;

4. Pourquoi ? Parce qu'ils n'ont pas eu le soutien d'Alfred, leur directeur, en termes de budget de formation et d'accompagnement ;

5. Pourquoi ? Parce qu'Alfred lui-même n'a pas compris ce qu'était l'agilité pour pouvoir soutenir ses équipes dans la démarche.

Ce soutien nécessaire aurait pu prendre plusieurs formes. Il fallait d'abord s'assurer qu'Alfred comprenne bien comment l'agilité vient répondre aux problèmes auxquels sa direction fait face, puis qu'il débloque les budgets requis pour soutenir le développement et ainsi assurer le succès de la démarche.

L'exemple ci-dessus, nous l'avons rencontré de nombreuses fois dans notre pratique professionnelle. Trop d'organisations croient que l'agilité est simplement une autre méthodologie et ils ne réalisent pas qu'elle impose souvent un changement culturel important. En fait, elle a aussi le pouvoir de transformer la méthode traditionnelle en la rendant plus agile lorsque les deux modes sont présents dans l'organisation (agile et traditionnel). Les organisations qui comprennent les fondements et les raisons

de l'agilité supportent et financent l'initiative de leurs équipes et ainsi récoltent les bénéfices recherchés.

13.1.2 Critères de succès pour les activités de soutien au développement

Le soutien au développement est inévitable pour assurer le succès de l'adoption de l'agilité. Ce soutien peut prendre diverses formes. Indépendamment de la forme choisie, certains critères de succès prévalent :

- Un **mécanisme efficace de diffusion et communication** des pratiques, façons de faire, processus, comportements attendus, leçons apprises et tout autre élément jugé pertinent pour le succès des projets.
- Une **présence sur le terrain** de ceux qui soutiennent le développement. Ces derniers ne doivent surtout pas être dans leur tour d'ivoire mais plutôt participer activement avec les équipes.
- Une **analyse objective des dérogations aux processus** pour favoriser les pratiques créatives et efficaces dans l'atteinte des objectifs.
- Ceux qui font le soutien au développement doivent aussi être **responsables de la mise en œuvre de l'agilité** et en être imputables auprès des gestionnaires ayant choisi l'agilité.
- L'**appropriation des méthodes agiles et pratiques d'ingénierie logicielle par les équipes de développement** devient l'indicateur de succès de l'adoption de l'agilité. L'agilité ne s'instaure pas toute seule, les équipes doivent être guidées pour y parvenir.

13.2 LES PRATIQUES DU SOUTIEN AU DÉVELOPPEMENT

De la mise en situation en début de chapitre, il est clair que l'équipe de soutien au développement a beaucoup de pain sur la planche si l'organisation est sérieuse dans son adoption de l'agilité et si tous partagent le même sentiment d'urgence face aux objectifs ciblés. Voici quelques exemples illustrant les premières activités de soutien au développement au début d'une transition agile :

- Donner des séances de formation sur l'agilité aux exécutifs et aux gestionnaires afin d'obtenir leur soutien.
- Déterminer le processus d'adhésion des projets aux nouvelles méthodes agiles pour maximiser les chances de succès de la démarche. Puis, élaborer et appliquer des critères d'évaluation pour identifier le risque d'amener une nouvelle équipe à adopter l'agilité.
- Sélectionner deux projets candidats pour appliquer l'agilité.
- Donner des séances de formations aux membres des premières équipes de développement afin qu'ils appliquent l'agilité correctement dès le départ.
- Accompagner les équipes dans le démarrage de leurs projets agiles.

- Faire des ateliers de développement du processus initial afin qu'ils servent de média à la communication des pratiques et livrables attendus.
- Participer aux premières rétrospectives des équipes de développement afin d'y récolter les améliorations à apporter au processus et pouvoir les diffuser aux autres équipes.
- Analyser les outils de gestion agiles du marché afin d'en sélectionner un convenable à l'organisation.

La portée des pratiques du soutien au développement peut être vaste, selon les besoins de montée en compétences et l'ampleur du groupe TI. Néanmoins, la suite de cette section contient les grandes activités faisant partie du soutien au développement, en faisant fi pour le moment de ceux qui les exécutent.

13.2.1 Prodiguer la formation continue sur les nouvelles pratiques

La première responsabilité des personnes assurant le soutien au développement est de monter en compétences les personnes impliquées dans les projets agiles, typiquement via des activités de formation pouvant prendre des formes variées : séminaires, séances de formation, ateliers de pratiques, accompagnement, lectures dirigées, etc. Il faut faire attention à former l'ensemble des personnes impactées par l'agilité, soit toutes celles intervenant avec les équipes agiles et ce à tous les niveaux de gestion.

Portée de la formation

Cette montée en compétences se fait sur les axes suivants :

- Les **pratiques agiles** : ces pratiques varient en fonction des méthodes agiles choisies et adaptées au contexte et à la culture de l'organisation. Les aspects théoriques sont certes abordés mais il faut mettre l'accent sur les aspects comportementaux des individus appliquant ces pratiques.
- Les **pratiques d'ingénierie logicielle** : ces pratiques varient en fonction des technologies adoptées et du niveau de qualité requis par les clients
- Les **processus de développement et d'évolution** et leurs différentes évolutions à travers le temps : les processus couvrent typiquement l'enchaînement des diverses pratiques dans un flux de travail et les artefacts qui en découlent, tel que les livrables, les livraisons de logiciel, les mesures et indicateurs et les leçons apprises ressortant des pratiques d'inspection et d'adaptation.
- Le **domaine d'affaires** : les équipes de développement dont les membres ont des connaissances du domaine d'affaires sont plus productives car elles éliminent du gaspillage en développant souvent du premier coup des solutions utiles qui plaisent au responsable de produit.
- Les **outils de gestion agile** retenus : gérer un projet agile avec un simple tableur ou des feuilles *Post-it*™ peut faire l'affaire dans un contexte d'un seul projet de quelques mois et avec seulement quelques personnes. Mais dès qu'il y a plusieurs équipes et des projets d'envergure, les limites d'un tableur sont rapidement atteintes et des outils spécialisés sont nécessaires. Le personnel doit apprendre

à s'en servir adéquatement et ces outils doivent être configurés au contexte de l'organisation (sécurité, disponibilité, fonctionnalités, paramètres et autres).

La formation ne doit pas seulement couvrir le savoir – soit les connaissances liées au sujet – mais aussi le savoir-faire – soit être capable d'appliquer les connaissances acquises dans tous les axes –, et le savoir-être, particulièrement en ce qui a trait aux pratiques agiles (Figure 13.1).

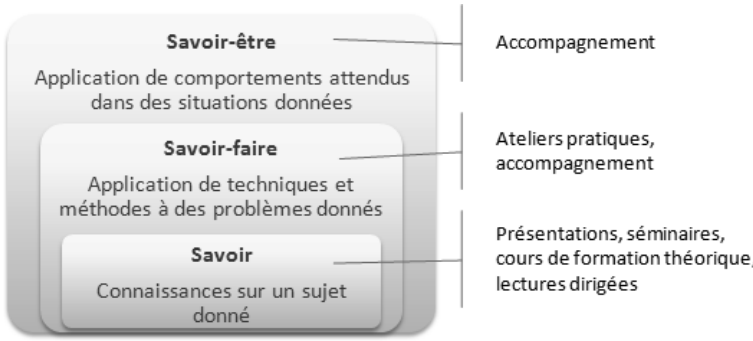


Figure 13.1 — Les trois niveaux de compétences visés par la formation et les mécanismes de formation associés.

Formation interne ou externe ?

Dans la plupart des organisations qui débutent avec l'agilité, il n'existe pas de personnes ayant les compétences d'un formateur expert en agilité. C'est pourquoi il est nécessaire de se tourner vers des ressources extérieures. Cependant, cette solution n'est pas souhaitable à long terme si l'organisation veut s'approprier le savoir-faire et le savoir-être agiles dans son contexte et sa culture. Il faut envisager rapidement de former des experts internes ayant des prédispositions à donner de la formation et à faire du *coaching*[1]. Ce sont ces personnes qui seront appelées à faire le soutien au développement après le départ des ressources externes et ainsi assurer la pérennité de l'initiative.

13.2.2 Coacher sur les pratiques agiles

Selon Jean-René Rousseau, un collègue coach agile, la mission d'un tel coach est « *d'aider les* ***organisations****, les* ***équipes*** *et les* ***individus*** *à atteindre des résultats exceptionnels via l'utilisation efficace des principes et pratiques agiles* »[2]. Adopter l'agilité pose des défis pour ces trois entités et requiert quatre niveaux d'intervention de la part des coaches (Figure 13.2).

1. *Coaching* et accompagnement sont synonymes.
2. Jean-René Rousseau, *Un coach agile : ça mange quoi en hiver ?*, présenté à Agile Montréal le 16 mars 2011.

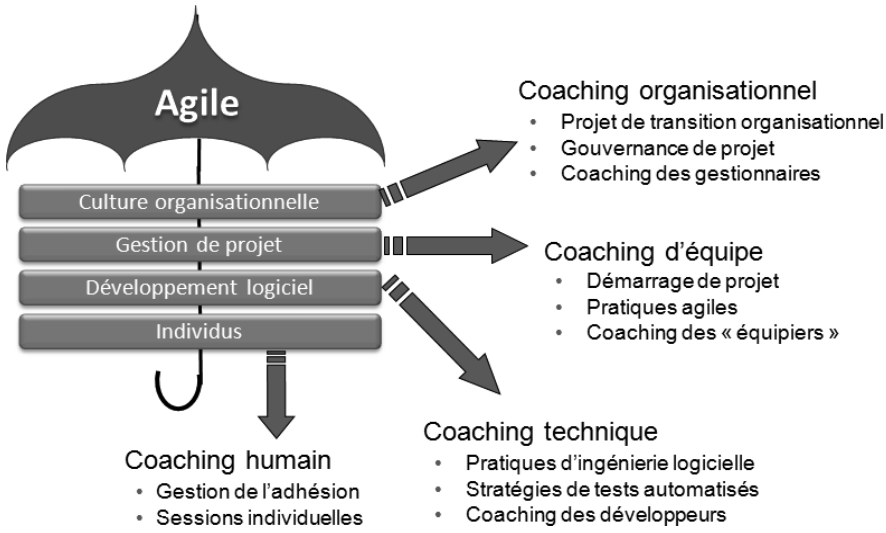


Figure 13.2 — Les quatre niveaux d'intervention du coaching agile.

Valeur agile : Les individus et les interactions davantage que les processus et les outils.

Le coaching est une forme importante d'éducation contribuant à la montée en compétences des membres des équipes. En effet, il contribue à transformer les connaissances acquises au cours des activités de formation en compétences pérennes de savoir-faire et de savoir-être. Cependant, ne devient pas coach qui veut, car plusieurs caractéristiques incombent au coach, notamment qu'il soit dépourvu d'autorité hiérarchique sur les personnes accompagnées, qu'il puisse adapter son style de leadership et qu'il puisse adapter sa posture de coach à la maturité et la cohésion évolutives de chaque équipe[1] (Figure 13.3).

Nous distinguons les différentes postures de coach avec l'approche d'apprentissage employée avec les personnes coachées :

- Le **formateur** décrit, explique et démontre son sujet, souvent de manière unidirectionnelle, de façon à augmenter le savoir des participants. Cette posture est nécessaire lorsque l'acquisition de connaissances est requise.
- L'**expert** décrit, explique et démontre son sujet en plus de donner des conseils, souvent de manière interactive avec les participants, de façon à augmenter leur savoir-faire. Cette posture est nécessaire lorsque l'acquisition de compétences est requise pour appliquer et maîtriser des techniques et des méthodes.
- Le **mentor** interagit avec son participant, surtout par des rencontres individuelles, où le participant expose ses difficultés et le mentor l'aiguille vers des

1. Se référer au chapitre *11 Gouvernance*, pour de détails sur l'évolution de la cohésion des équipes.

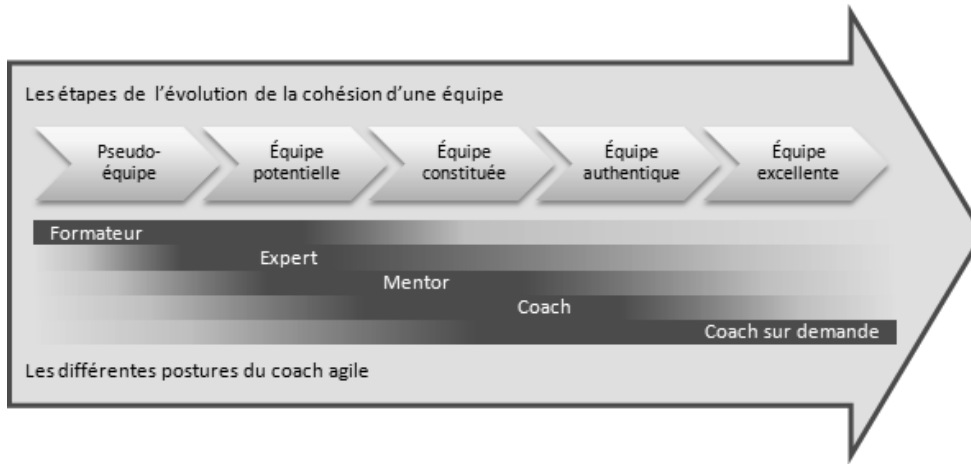


Figure 13.3 — Les postures de coach à adapter au niveau de maturité de chaque équipe[a].

a. Source pour les étapes de l'évolution de la cohésion des équipes : Oliver Devillard, *Dynamiques d'équipes*, Éditions d'organisation, 2005. Source pour les postures de coach : Pierluigi Pugliese, *Solution Focused Approach to Agile Coaching*, Connexxo, http://www.slideshare.net/ppugliese/solution-focused-approach-to-agile-coaching-2604381, consulté en mars 2011.

solutions. Cette posture pourrait s'avérer nécessaire à tout moment et vise surtout l'augmentation du savoir-faire et parfois celle du savoir-être.

- Le **coach** questionne ses participants pour les faire réfléchir et les amener à trouver eux-mêmes les réponses de façon à ce que les notions apprises soient plus vite maîtrisées. Le coach se doit de montrer les conséquences aux solutions mises en œuvre ou envisagées par ses participants. L'autogestion des équipes implique de les laisser prendre leurs décisions mais en les guidant dans des cycles d'apprentissage le plus court possible et en les accompagnant dans l'inspection et l'adaptation de leurs pratiques basées sur les conséquences vécues, bonnes ou mauvaises. Cette posture vise l'augmentation du savoir-être des participants de façon à ce qu'ils aient les comportements attendus dans des contextes donnés, et vise l'augmentation de la cohésion de l'équipe.
- Le **coach sur demande** est un coach qui n'a besoin d'intervenir qu'à quelques occasions au lieu d'être présent à plein-temps avec l'équipe, car cette dernière est hautement cohésive et autonome.

13.2.3 Gérer le processus

Une des pratiques importante du soutien au développement est la gestion du processus. Cette activité comprend la documentation du processus, son évolution via les mécanismes d'amélioration continue et la diffusion de ses éléments changeants. Les pratiques relatives à la gestion du processus, soit la documentation, l'entretien et l'évolution, ainsi que la diffusion sont décrites au chapitre *12 Évolution du processus*.

Mais la gestion du processus signifie aussi la surveillance de son application. En mode agile, cette pratique peut être différente de celle mise en œuvre pour les processus traditionnels. En effet, nous avons souvent vu des processus traditionnels détaillés dont la surveillance consistait parfois à s'assurer que chaque livrable et chaque pratique est effectivement appliquée dans l'ensemble des projets[1]. En mode agile, il faut voir à ce que le processus appliqué mette l'accent sur le respect des objectifs des pratiques et des livrables plus que sur les moyens choisis par l'équipe pour y parvenir. Cette façon de faire favorise une émergence de moyens créatifs et efficaces au sein des équipes pour appliquer l'une ou l'autre des pratiques agiles, pouvant être récupérés par les responsables de la gestion du processus pour servir d'exemples et en inspirer d'autres lorsque les résultats se sont avérés concluants. Toutefois, il pourrait être nécessaire d'imposer certains livrables ou certaines pratiques au démarrage des équipes lorsque l'évolution de la cohésion ne dépasse pas l'équipe potentielle afin d'assurer une mise en œuvre minimale des valeurs et principes agiles.

Diffuser et communiquer les changements aux pratiques

Des améliorations aux processus vont survenir régulièrement car le processus est rendu vivant par les pratiques d'inspection et d'adaptation des différentes méthodes agiles. Ces améliorations deviennent potentiellement utiles à n'importe quel projet et des moyens de diffusion doivent être mis en œuvre.

Un de ces moyens peut être les « communautés de pratiques » qui regroupent des personnes dont les rôles et les techniques appliquées sont similaires (programmeurs Java, Scrum Masters ou coach d'équipe, architectes, etc.). Ces communautés se rencontrent régulièrement (d'une à quatre fois par mois) afin de partager les problèmes rencontrés et les solutions trouvées. C'est aussi une occasion pour soumettre un problème à sa communauté et voir si quelqu'un a une solution. Ainsi, la communauté entière participe, en quelques minutes, à élaborer une solution. C'est une application concrète du *cerveau collectif*[2] que nous avons trouvée très efficace lorsque nous y avons pris part.

Les *newsletters* périodiques sur les améliorations aux processus peuvent informer en comblant le volet du savoir mais elles n'augmentent pas le savoir-faire ni le savoir-être. Par contre, sachant qu'une nouvelle façon de faire est apparue récemment, un membre d'équipe sera plus à même de réclamer de l'accompagnement ciblé lorsque viendra le temps de la mettre en œuvre.

Rapporter les mesures de performance du processus

Étant donné que la gestion du processus fait partie des tâches du soutien au développement, par extension, nous proposons que ce soit le même groupe ou les mêmes personnes qui maintiennent les mesures de performance du processus et rapportent les

1. Syndrome de la « taille unique » discuté au chapitre *14 Agilité et documentation*.
2. Fondé sur le proverbe disant « *deux têtes valent mieux qu'une* », le *cerveau collectif* se produit quand l'ensemble des personnes participent à faire émerger des idées et des solutions.

résultats au propriétaire des processus ou à la direction de l'organisation. Les mesures de performance du processus sont traitées en détail au chapitre *15 Mesures de performance*.

13.2.4 Obtenir des ressources aux équipes

> **Principe agile** : Bâtissez le projet autour de personnes motivées. Donnez-leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail.

Les personnes faisant le soutien au développement peuvent travailler à obtenir des ressources pour les équipes agiles, de façon à isoler ces dernières des préoccupations matérielles et de logistique. Ces ressources sont principalement liées aux **locaux et à l'environnement de travail**. Nous en présentons une liste non exhaustive ci-après.

Pouvoir rassembler les membres d'équipe

Il est recommandé de co-localiser les membres d'une équipe agile, c'est-à-dire non seulement de les mettre tous à proximité, mais aussi que les parois de leurs espaces de travail soient retirées ou abaissées de façon à ce qu'ils puissent se voir et s'entendre à plein-temps, augmentant ainsi leur efficacité en favorisant la communication. Nous avons rencontré plusieurs organisations où cela s'est avéré compliqué. Il y a plus d'avantages que d'inconvénients à abattre les parois entre les bureaux, surtout si elles sont mobiles.

Pouvoir utiliser les murs pour coller de l'information

Beaucoup d'équipes agiles travaillent avec des feuilles *Post-it*[tm] ou des petits cartons qu'elles collent au mur afin de rendre l'information visible à tous. Mais encore faut-il que ces équipes obtiennent le droit de coller des choses sur les murs. Une équipe appliquant la méthode *Kanban* doit obtenir ce droit car son outil de travail principal est un tableau sur lequel est collé un petit carton illustrant chaque tâche.

Avoir des tableaux blancs et une caméra numérique

C'est bien connu, les équipes agiles font de l'architecture et de la conception avec un tableau blanc et des crayons feutres secs[1]. Les membres d'équipes y travaillent ensemble et, pour se rappeler du contenu, ils prennent une photo numérique des résultats qu'ils veulent conserver et placent cette photo sur le serveur, dans un endroit approprié et accessible par tous. Cette technique efficace suppose que l'organisation a fourni le tableau, les crayons feutres et la caméra. Cette dépense de quelques centaines d'euros ou de dollars est vite rentabilisée par les gains en efficacité.

1. Des ardoises murales et des craies peuvent faire l'affaire si vous les avez déjà mais ça crée des poussières désagréables qui collent aux doigts et aux vêtements.

Avoir des serveurs de tests et d'intégration et leurs droits d'accès associés

La pratique de l'intégration continue est essentielle pour assurer le succès d'une équipe agile. Pour y parvenir, les équipes ont besoin de serveurs d'intégration et de tests sur lesquels les tests automatisés s'exécuteront à chaque mise à jour du code : au minimum une fois par jour, de préférence à chaque soumission de code. Lorsque l'organisation ne fournit pas d'ordinateurs supplémentaires, les développeurs ont tendance à en monter un sur l'un de leurs postes. Cela les rend inefficaces et ils gaspillent du temps précieux de développement alors que l'organisation n'a qu'à leur fournir un ordinateur fonctionnel et configuré pour couvrir leurs besoins. L'équipe de développement doit avoir tous les droits d'accès requis sur des environnements pour installer ses composantes.

Avoir une salle de rencontre pour y tenir les mêlées quotidiennes

Les équipes agiles font en général une brève rencontre tous les jours pour coordonner les activités du projet. Il est préférable qu'elles obtiennent une salle à cet effet pour ne pas déranger les autres équipes à proximité.

Avoir accès à une salle de grande capacité

Il peut être requis, dans le cas de grands projets (par exemple, plus de 50 personnes), d'avoir accès à une salle de grande capacité pour les revues d'itération ou autres rencontres de groupes. Lors de jalons importants, comme la livraison d'une version attendue, il peut être bénéfique de réunir tous les intervenants – développeurs et utilisateurs – dans une même grande salle afin de démontrer les fonctionnalités développées.

13.3 MISE EN ŒUVRE D'UN GROUPE DE SOUTIEN AU DÉVELOPPEMENT

Dans toute transition agile, il existe des pratiques de soutien au développement dont le degré de formalisme peut varier (Figure 13.4) selon les besoins, le budget alloué ou la compréhension des dirigeants de l'impact de l'agilité sur les projets et l'organisation.

Avant de former une équipe dédiée de soutien au développement, il est recommandé d'établir la liste des besoins en tenant compte du nombre et de la taille des équipes agiles et du nombre de clients, internes ou externes, qui seront appelés à travailler au sein de ces équipes. De plus, une analyse des coûts et des bénéfices anticipés est de mise pour déterminer quel mode de fonctionnement est le plus approprié à la taille de l'organisation. Le choix n'est ni blanc ni noir : des formes hybrides de celles illustrées à la Figure 13.4 peuvent être adoptées.

Le fait de formaliser une telle équipe permet d'en contrôler les coûts et la rentabilité puisque leur apport est censé permettre aux équipes de développement d'atteindre leurs objectifs plus efficacement.

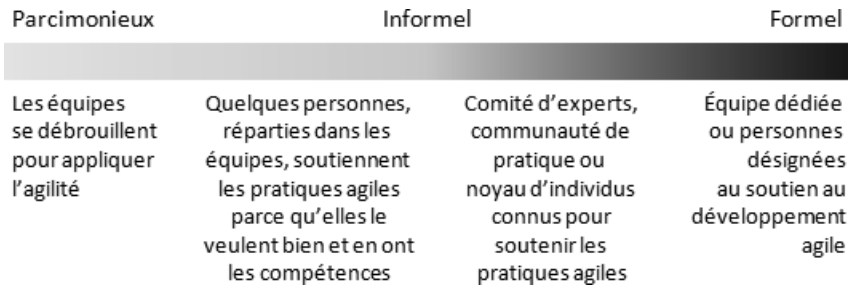


Figure 13.4 — Degrés de formalisme des activités de soutien au développement pendant une transition agile.

13.3.1 Les formes d'un groupe de soutien au développement

Quelques personnes, réparties dans les équipes

Cette façon informelle de soutenir le développement s'adapte bien aux petites organisations ou aux organisations débutant leur transition agile avec quelques projets pilotes. Les coaches agiles faisant du soutien au développement dans ce mode sont souvent apparentés à des Scrum Masters lorsque la méthode Scrum est adoptée. Mais ils peuvent également avoir des rôles différents tels qu'architecte ou développeur senior. Ils sont répartis dans chacune des équipes et encadrent les pratiques agiles et d'ingénierie logicielle en tant qu'expert relevant de la même personne que le reste des membres de l'équipe.

Cette façon de faire a ses limites dans une transition agile affectant un grand nombre de projets où la maîtrise des processus organisationnels est requise pour des besoins d'affaires tels que la conformité à des normes, lois et règlements ou des besoins de contrôle des coûts liés aux processus. Pour contrer ces limites, une mise en commun et un partage des connaissances et compétences de ces experts sont requis.

Comité d'experts, communauté de pratique ou noyau d'individus

Le comité d'experts est une évolution naturelle de la mise en œuvre par des experts répartis dans les équipes. Ces mêmes experts se regroupent en communauté à intervalles réguliers pour partager leurs connaissances, compétences et problématiques vécues. Le comité est alors en mesure de soutenir chacun des membres de sa communauté vivant des difficultés ou ayant à surmonter des défis dans la mise en œuvre de certaines pratiques agiles. De plus, certaines personnes pouvant avoir leurs spécialités propres, il devient utile de les faire intervenir dans une équipe rencontrant un besoin qu'elles peuvent combler par une intervention ponctuelle.

Pour que ce comité fonctionne bien, il est utile de rédiger une charte d'équipe, semblable à une charte de projet mais non limitée dans le temps, avec une vision, des objectifs, des règles de fonctionnement et la liste des membres et leurs spécialités respectives au besoin. Cette charte doit rester vivante et l'un des membres du comité doit prendre le leadership du groupe pour faire le suivi de l'atteinte des objectifs et du respect des règles.

Ce type de mise en œuvre fonctionne dans des petites et moyennes organisations, voire dans des grandes, jusqu'à ce que les tâches de normalisation ou de gestion du processus accaparent certains des membres à un point où ils deviennent inefficaces dans leur rôle de coach par manque de disponibilité.

Équipe autonome et dédiée

Lorsque la taille de l'organisation devient importante, il devient naturel de constituer une équipe autonome et dédiée pour faire le soutien au développement. Cette équipe devient alors le centre d'expertise dont la mission principale est de mener l'ensemble des activités de soutien au développement citées à la section précédente.

13.3.2 Insertion du soutien au développement dans une structure existante

Dans certaines grandes organisations, les dirigeants ont préféré ne pas créer de structure indépendante pour une équipe de soutien au développement mais d'adjoindre les personnes responsables de ce volet dans une équipe existante, tel que le bureau de projet, l'équipe de méthodologie, le groupe d'assurance qualité ou le bureau d'amélioration continue. Ce qui importe, c'est que les personnes auxquelles le soutien au développement est attribué appliquent leurs pratiques et les critères de succès avec les équipes agiles. Par exemple, ces personnes doivent aller trouver les équipes et les aider à appliquer et adapter le processus. **Il ne faut pas s'attendre à ce que ce soit les équipes qui viennent à elles sinon le processus ne sera pas appliqué.**

13.3.3 Taille requise d'un groupe de soutien au développement

Selon notre expérience, le coût des activités de soutien au développement devrait osciller entre 3 % et 5 % de l'effort annuel de développement[1]. Il est possible que ce soit plus élevé la première année, entre 4 % et 10 %, le temps que les membres d'équipes des projets pilotes fassent leur montée en compétences et complètent avec succès leurs premiers projets. Tout dépend du niveau de compétences initial des membres d'équipes et du niveau de compétences souhaité.

Il ne faut pas s'attendre à ce que le coût du soutien au développement devienne nul avec le temps. Ce n'est pas possible puisqu'un processus en amélioration continue requiert des efforts pour être mis à jour et communiqué. De plus, étant donné les changements technologiques fréquents (nouvelle plateforme, nouvel outil de développement, ou nouvelles versions des outils utilisés), il est normal de vouloir soutenir les développeurs à travers tous ces changements.

1. C'est également ce que concluait le *Software Engineering Institute* pour l'ensemble des pratiques liées au processus, peu importe que l'organisation soit agile ou pas.

13.3.4 Pièges à éviter

Niveau d'imputabilité inadéquat

Dans l'organisation, un groupe formel de soutien au développement doit relever d'un niveau hiérarchique avec une autorité non contestée et non contestable. Un manque d'autorité et d'imputabilité du groupe de soutien au développement risque de diminuer l'écoute des intervenants mettant en œuvre les méthodes et la philosophie de travail. On peut facilement imaginer que dans l'urgence, on préfère obéir à la hiérarchie que d'appliquer les nouvelles méthodes.

Capacité à desservir les équipes

Pendant la transition agile, l'organisation doit respecter la capacité de l'équipe de soutien au développement lorsque vient le temps d'augmenter le nombre de projets agiles en démarrage et en cours, car ces projets requièrent de l'accompagnement. Cela devient une condition de succès pour la qualité de la mise en œuvre de l'agilité.

Réinventer le matériel de formation

Il est souvent moins long et moins coûteux d'obtenir les droits de diffusion de matériel de formation existant que de le refaire de toutes pièces. Il est alors préférable de limiter la création de nouveau matériel à ce qui est spécifique à l'organisation.

Dictature des méthodologistes

Les personnes travaillant en soutien au développement à plein-temps sont souvent appelées des « méthodologistes » puisque leur travail consiste à gérer, communiquer et assurer l'applicabilité et la qualité des divers éléments des méthodologies en place. Lorsque les méthodologistes se comportent en coach avec les membres des équipes, en les accompagnant dans leur montée en compétences, en les guidant dans leurs choix de mise en œuvre de certaines pratiques et en leur faisant comprendre l'impact d'une mauvaise application d'une pratique, alors tout va pour le mieux. Mais ce n'est pas toujours le cas. Certains méthodologistes se croient investis d'une autorité sur les membres des équipes et se comportent en « police des processus » en blâmant toute personne appliquant mal ou pas du tout une pratique faisant partie du processus. Une attitude dogmatique et intransigeante face au processus n'est certainement pas souhaitée si on applique les concepts de l'amélioration continue.

> Blague d'un développeur entendue dans une grande société non agile : « *Quelle est la différence entre un terroriste et un méthodologiste ? Réponse : il est possible de négocier avec un terroriste !* »

Pour éviter cette situation désagréable, il est important que les comportements attendus des personnes faisant le soutien au développement soient clairement énoncés. Ainsi, lorsque le processus n'est pas suivi, il est important de comprendre pourquoi, et de traiter le problème à la source sans jamais blâmer qui que soit. Sinon, l'équipe du soutien au développement ne sera jamais sollicitée par les individus de l'organisation. C'est leur savoir-être qui prime ici.

Conseil : faire connaître les pratiques obligatoires afin de faire respecter les contraintes de gouvernance. Ces contraintes doivent être connues des équipes pour les aider à prendre des décisions.

Affecter des personnes dont la crédibilité est insuffisante

Les tâches de soutien au développement s'accomplissent par des experts, pas par des débutants. Toutefois, un expert sur un sujet donné n'est pas nécessairement expert sur tout. Par exemple, un expert en gestion traditionnelle de projet peut difficilement être déclaré expert en gestion agile de projet s'il n'a suivi que deux jours de formation sans vraiment les avoir mis en pratique. Pour avoir une crédibilité suffisante, les personnes affectées au soutien au développement doivent avoir expérimenté les pratiques qu'ils coachent. Sinon, vous aurez un cas où c'est « un aveugle guidant l'aveugle » et vous risquez de ne pas bénéficier des retombées souhaitées.

En résumé

L'agilité ne se met pas en place par la seule volonté des membres des équipes. Dans tous les cas, il est nécessaire de leur fournir de la formation et de l'accompagnement. Plus la taille du groupe de développement logiciel est grosse, plus il faut de communication, collaboration et encadrement sur les processus, les rôles et responsabilités, les livrables attendus et les améliorations apportées aux façons de faire. Une personne ou un groupe de personnes doivent mettre en œuvre ces responsabilités et servir de catalyseurs pour les équipes de développement.

14

Agilité et documentation

Objectif

Dans ce chapitre, vous découvrirez que la documentation est essentielle dans les projets agiles, contrairement aux préjugés qui prétendent qu'elle est quasi inexistante avec une approche agile. Vous aiguiserez votre sens critique au sujet de la qualité, de la quantité et de la forme des documents à produire et vous découvrirez quelques stratégies et techniques pour aider à la rédaction de la documentation en équipe.

14.1 LES BESOINS LIÉS À LA DOCUMENTATION

14.1.1 Comprendre ses besoins en information

> **Valeur agile :** Les logiciels fonctionnels davantage qu'une documentation exhaustive.

Cet énoncé est probablement la valeur du *Manifeste agile* la plus souvent mal interprétée. Elle peut laisser croire que les praticiens des méthodes agiles n'accordent aucune importance à la documentation.

Pour les adeptes de la rédaction, cette valeur semble menacer le sentiment de sécurité apporté par la documentation. Il peut être rassurant de savoir que l'information d'une organisation est disponible et accessible pour consultation. Pour les réfractaires à la documentation, cette valeur est une bénédiction, car elle représente l'opportunité de se consacrer à n'importe quelle activité autre que celle de la rédaction.

Cependant, le *Manifeste agile* ne propose absolument pas l'élimination de la documentation. Il rappelle plutôt que bien que la documentation soit utile et nécessaire,

elle ne remplacera jamais l'objectif principal d'un projet de développement, soit la livraison d'une solution logicielle fonctionnelle. La documentation n'est pas une fin, elle est un moyen d'y parvenir lorsque les besoins sont connus. Par conséquent, il est important de s'assurer que les documents livrés répondent aux besoins et qu'ils sont utiles aux différents consommateurs (Figure 14.1).

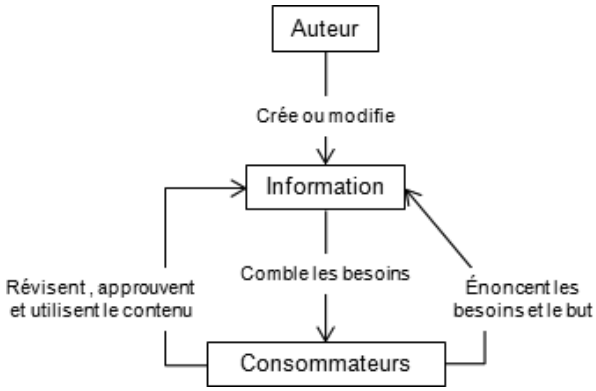


Figure 14.1 — Modèle simplifié lié à l'information.

Retour d'expérience : une réduction de la documentation

Le diagnostic des pratiques de développement d'une société d'assurance avait fait ressortir plusieurs problèmes liés aux dossiers d'analyse fonctionnelle :

– Taille énorme de 50 à 250 pages par fonctionnalité ;

– Redondance dans le contenu, notamment sur les descriptions détaillées des données en entrée, sortie, lecture ou écriture ;

– Vocabulaire inadéquat pour le lecteur ciblé, comme des informations de conception détaillée s'adressant à des spécialistes de l'assurance.

J'ai mené un atelier de révision du gabarit avec les analystes fonctionnels et le pilote du système pour résoudre les problèmes identifiés. En seulement trois heures, nous avons réussi à réduire la taille de moitié, éliminer les redondances et ranger les informations de conception en annexe.

Sylvie

14.1.2 La documentation : un outil de communication

La documentation doit être vue comme un moyen de communiquer de l'information, de faire connaître ce qui est utile et nécessaire à tous ceux auxquels cette information est destinée, tout en tenant compte du contexte dans lequel elle sera utilisée. Ce contexte est directement lié à l'usage visé par la documentation :

- **Matériel de formation** : que ce soit pour les futurs membres d'équipe ou pour les utilisateurs devant développer leurs connaissances et compétences sur un sujet donné.

- **Soutien à des discussions à venir** : parce que toutes les personnes ne sont pas des communicateurs nés, il peut être utile de consigner l'objet de la future discussion par écrit, ainsi qu'une liste de points principaux. Cela permet de bien se préparer à ces discussions tout en demeurant concentré sur le sujet dont il est question. Même quand une personne a accès facilement à l'auteur de l'information, cette documentation permet d'éviter de la répéter, rendant les discussions plus efficaces et profondes par les personnes qui en ont pris connaissance.
- **Décisions prises lors de discussions** : parce qu'il est nécessaire de laisser des traces et ainsi limiter la dispersion. Ces traces sont utiles au terme d'une rencontre afin de communiquer les décisions et les raisons ayant alimenté cette prise de décision[1]. Elles sont aussi utiles pour la gestion de projet où l'exemple le plus probant est sans aucun doute la charte de projet, visant à recentrer les membres d'équipe sur les objectifs et les conditions de succès au cas où ils s'égareraient.
- **Communication à un large public** : étant donné qu'il n'est pas possible ni efficace de réunir toutes les parties intéressées dans une discussion, il devient utile et nécessaire de consigner certaines informations destinées à l'ensemble de ces personnes. C'est le cas, par exemple, des bilans d'itérations et des bilans de projets.
- **Proximité des intéressés** : l'information peut ne pas être très détaillée par son auteur sachant que ceux auxquels elle est destinée sont tout près de lui et que cette documentation sert de support aux discussions, tel que les scénarios utilisateurs (*user stories*).
- **Matériel de référence** : un jour viendra où l'équipe de projet cessera d'exister et où l'application développée sera transférée à l'équipe de maintenance. Certaines informations dites « de référence » deviennent alors utiles et nécessaires, tel que la recette pour produire une version exécutable de l'application.

Nonobstant ces contextes, nous croyons que trois facteurs sont à considérer lorsque vient le temps d'évaluer les besoins de documentation, soit les consommateurs de l'information, la qualité et le gaspillage potentiel.

Les consommateurs de l'information et leurs besoins

Il importe donc de connaître l'auditoire auquel l'information est destinée :

- **Le consommateur ciblé** : Lorsqu'un document est destiné un large public, il devient difficile de couvrir les besoins de tous et de chacun. En ne précisant pas à qui s'adresse l'information, elle risque d'être abondante et diluée, réduisant l'efficacité de sa consultation. Pour fixer les attentes, une bonne pratique est d'indiquer, au départ, le consommateur auquel l'information est destinée.

1. La forme peut être légère, comme la photo du tableau blanc sur lequel les informations ont été notées ou une cartographie conceptuelle (*MindMap*).

- **Le profil du consommateur** : Il faut s'assurer que le niveau de documentation est accessible en termes de compréhension pour le consommateur ciblé. Dans la mesure du possible, il faut s'adresser à lui dans son langage et traiter des sujets dans leurs contextes. Il est une bonne pratique de fixer les attentes envers les lecteurs lorsque des connaissances particulières sont nécessaires pour comprendre la documentation, par exemple en indiquant les documents préalables à la compréhension ou encore le niveau de connaissance requis pour comprendre le contenu. Par exemple, si le client doit approuver un dossier d'analyse, il est attendu que ce dossier soit rédigé en utilisant le vocabulaire d'affaires et qu'il ne contient pas d'information de conception que seuls les programmeurs comprennent. Autrement, cela met le client dans l'embarras devant son incompréhension du contenu d'un document lui étant pourtant destiné.
- **Le but de l'information** : Si possible, il est préférable de diviser les différentes considérations dans différentes sections, voire différents documents. Avec un référencement et un titre éloquent, la recherche de l'information est plus efficace. Répéter le titre dans le but est à proscrire[1]. À la place, il faut décrire l'usage que le consommateur fera de cette information. Une ou deux phrases courtes sont suffisantes en général. Voici un exemple de mauvais but pour ce document dont le titre est « *Dossier d'analyse fonctionnelle du paiement d'achats en ligne* » serait « *Le but de ce document est de décrire le paiement d'achats en ligne* ». Il est préférable que le but soit « *Ce dossier s'adresse au pilote du système afin qu'il valide les règles d'affaires et le comportement souhaité de la fonctionnalité de paiement en ligne. Ce dossier sert également aux testeurs qui définissent la liste complète des tests à effectuer.* ».

Les trois « U » de la qualité

Au cours de nos interventions, nous faisons référence aux trois « U » de la qualité de l'information, soit :

- **Utile** : l'information est pertinente pour son consommateur et le contenu comble ses besoins.
- **Utilisable** : la documentation est structurée pour en faciliter l'usage et elle est référencée de manière à faciliter la recherche d'information. Son contenu est juste, à jour et organisé.
- **Utilisée** : l'information est effectivement utilisée par les consommateurs pour atteindre le but recherché.

Il faut porter une attention particulière à la qualité et l'efficience de l'information plutôt qu'à sa quantité.

1. Nous avons maintes fois constaté ce gaspillage au cours de nos interventions.

Le gaspillage potentiel

Les formes de gaspillage décrites par la méthode *Lean* peuvent servir à identifier les éléments de gaspillage de la documentation. Le Tableau 14.1 permet d'en évaluer le gaspillage potentiel.

Tableau 14.1 — Les sept formes de gaspillage transposées au contexte de la documentation.

Gaspillage dans un contexte de développement logiciel[a]	Évaluation du gaspillage dans la documentation
1. Travaux incomplets	1. Combien de documents sont entamés mais pas encore disponibles aux utilisateurs, parce qu'ils sont incomplets, inutiles ou inutilisables ?
2. Caractéristiques ou fonctionnalités superflues	2. Y a-t-il de la documentation anticipant le développement de fonctionnalités non priorisées ? Y a-t-il du contenu redondant à l'intérieur d'un document ou d'un document à l'autre ? Y a-t-il des portions de la documentation peu ou pas utilisées ?
3. Réapprentissage	3. La fréquence et le délai requis pour mettre à jour la documentation est-il adéquat ? L'entretien de la documentation requiert-elle un effort déraisonnable compte tenu de la durée de vie utile de l'information ?
4. Permutation entre les tâches	4. Les travaux de rédaction sont-ils continus ou fréquemment interrompus par d'autres tâches incombant aux rédacteurs ?
5. Transferts (de tâches, de composantes)	5. Y a-t-il trop d'intervenants différents participant aux activités de correction et de validation, allongeant ainsi le temps requis pour publier le document (exemple d'intervenants : un rédacteur, un relecteur, un approbateur et des processus de déploiement) ? La documentation est-elle disponible et accessible par tous ses consommateurs ciblés ?
6. Délais	6. Le délai requis pour qu'une nouvelle information soit documentée et diffusée est-il adéquat ? Est-il possible d'accéder rapidement à l'information cherchée (quelques secondes versus plusieurs minutes ou pire, des heures) ?
7. Défauts et corrections	7. La documentation est-elle juste et mise à jour ? La qualité de la documentation est-elle jugée satisfaisante ?

a. Tiré et traduit de *Lean Software Development: An Agile Toolkit*, par Mary et Tom Poppendieck, Addison Wesley, 2003. Les sources de gaspillages dans un contexte de développement logiciel sont décrites plus en détails au chapitre *8 Infrastructure technologique*.

Conseil : *Une information est complète non pas lorsqu'on ne peut plus rien y ajouter, mais lorsqu'on ne peut plus rien en retirer*. Cette phrase est à garder en tête chaque fois que vous serez tentés d'enrichir une documentation, et que ces ajouts pourraient ne pas combler les besoins en information du consommateur.

14.1.3 Le syndrome de la « taille unique »

Devant une quantité importante d'informations, plusieurs organisations choisissent de les regrouper dans des documents dont le contenu est normalisé pour l'ensemble des projets sous un gabarit officiel. Ces gabarits, surtout l'usage qui en découle, ont évolué au cours des années, menant à des documents volumineux ne répondant parfois plus aux besoins de tous leurs consommateurs. C'est souvent le cas lorsqu'un seul gabarit existe pour un type de document donné – par exemple, les exigences ou spécifications – exempt de guide d'adaptation aux besoins spécifiques d'un projet ou d'une application. Ainsi, cet unique gabarit est une analogie au concept de « taille unique » pour certains vêtements, ne convenant pas à toutes les personnes.

Le seul avantage d'un gabarit unique est de simplifier la vie des responsables des actifs du processus. S'il est trop rigide, il devient inefficace car il impose l'obligation de documenter de l'information, souvent sous une forme spécifique, ne comblant peut-être pas les besoins des consommateurs de l'information. S'il contient des règles d'adaptation claires et simples, alors il peut être utile à plus large échelle.

Conseil : il est souvent utile de revoir l'application d'un gabarit au moment de son utilisation et d'enrichir ce dernier de règles d'adaptation pour le rendre plus facilement utilisable. La règle des trois « U » s'applique aussi aux artefacts de processus.

14.2 LES FORMES DE DOCUMENTATION

Lorsqu'une personne parle de documentation, elle fait souvent référence à des documents imprimés ou imprimables, contenant des informations à communiquer sous des formes variées telles que texte, tableau, graphiques et diagrammes. L'un des principes fondamentaux de la communication est de mettre en forme le message à transmettre pour que le consommateur le comprenne bien. Un format inadéquat a pour conséquence que l'information ne sera pas utilisée. Cette situation peut être causée, entre autres, par un outil non normalisé, trop de texte superflu ou un mauvais mécanisme de stockage.

Dans la majorité des sociétés où nous sommes intervenus, la majeure partie de la documentation était dans des fichiers textes ou dans des tableurs. Dans ces cas, la création et la mise à jour de l'information peuvent devenir fastidieuses, particulièrement si plusieurs personnes contribuent à chacun des documents, car les outils servant à les documenter ne permettent pas la modification simultanée. Des modifications sont alors perdues quand deux personnes appliquent des modifications en même temps, ce qui devient du **gaspillage**. Pour contourner cette irritante situation, les organisations nomment un responsable par document et toutes les modifications sont canalisées vers cette personne qui devient alors un goulot d'étranglement.

Mais est-ce que la forme de la documentation permet bien de communiquer efficacement de l'information à jour ? Pas forcément. Plusieurs critères doivent être

considérés pour déterminer la forme adéquate que peut prendre une information, tout en évitant le syndrome de la « taille unique » :

- **Volatilité** de l'information : une information temporaire ne doit pas requérir trop d'effort pour être mise « au propre » dans un outil sophistiqué ;
- **Disponibilité** de l'information : la forme donnée à l'information doit pouvoir être utilisée par ses consommateurs via les outils dont ils disposent ;
- **Conformité** à des normes, lois et règlements : le format de l'information ne doit pas contrevenir à des normes, lois ou règlements en vigueur ;
- **Unicité** de la source des informations : chaque information ne devrait provenir que d'une source unique, éliminant ainsi les redondances et favorisant l'efficacité.

Outre les fichiers textes ou tableurs, plusieurs formes de documentation sont à considérer quand un besoin en information est identifié :

- **Photos numériques** d'un tableau blanc sur lequel l'information est écrite : « une image vaut mille mots », dit le proverbe, et cela demeure vrai en développement logiciel. Cette forme de documentation est particulièrement utile pour capturer des décisions de conception ou de modélisation faites en équipe autour d'un tableau blanc. Chaque participant se souviendra des éléments discutés en visionnant la photo. Ces photos sont stockables en format électronique dans l'espace du projet ou de l'application. Cette forme nécessite la disponibilité d'une caméra numérique pour l'équipe de développement.
- **Cartographie conceptuelle** : économique et très utile pour organiser un grand nombre d'informations sur une seule page car cette cartographie conceptuelle requiert peu d'effort à créer et à modifier. Cette forme nécessite un outil spécialisé dont plusieurs sont disponibles gratuitement sur Internet[1].
- **Intranet** ou **pages wiki**[2] : utile pour consulter de l'information évoluant rapidement et en collaboration par plusieurs individus, tel que conception et exigences de produits et processus de développement. Cette forme nécessite d'installer un site wiki ou un intranet sur le réseau (généralement gratuit) dont il faut ensuite gérer les droits d'accès pour les ajouts et les modifications de contenus. On peut accéder aux wikis ou intranets via un navigateur Internet.
- **Présentations (diaporamas)** : utile pour de l'information résumée où texte et graphiques animés s'entremêlent pour permettre la communication efficace.
- **Les tests automatisés** : les développeurs praticiens de l'agilité sont de plus en plus nombreux à utiliser les tests automatisés comme forme de documentation. En utilisant un niveau de langage adéquat, un test permet d'indiquer les résultats et le comportement attendus par une fonctionnalité. L'écriture des

1. Notamment FreeMind (http://freemind.sourceforge.net/wiki/index.php/Main_Page) et XMind (http://www.xmind.net/), tous deux disponibles en français à l'exécution.

2. Un **wiki** est un site web dont les pages comportent des hyperliens les unes vers les autres et sont modifiables par les visiteurs autorisés afin de permettre l'écriture et l'illustration collaboratives des documents numériques qu'il contient – source : Wikipedia.org.

tests sous forme de code permet d'essayer la méthode en plus de la spécifier. Avec l'intégration continue, l'exécution des tests automatisés permet aux développeurs de prévenir l'introduction d'une erreur suite à une modification au code, ce qu'un test textuel ne peut pas faire.

- **Commentaires structurés dans le code** : il est rare qu'une documentation de la conception d'un système soit mise à jour à chaque modification du logiciel. La majorité des mainteneurs disent qu'ils préfèrent se fier au code et aux tests automatisés car ces derniers sont « vrais ». La documentation de conception devient vite désuète même en étant récente (gaspillage notoire en développement logiciel). Cependant, en l'absence de tests automatisés, la documentation textuelle reste la meilleure option.

La majorité des langages modernes de programmation permettent l'intégration d'outils de commentaires structurés directement dans le code[1]. Cette forme d'information est alors plus facilement mise à jour par les développeurs et les mainteneurs en plus de pouvoir faire l'objet de revue par les pairs. De plus, elle peut être extraite automatiquement lors de la construction de l'exécutable et rendue disponible via un intranet de produit.

Nous avons utilisé ces formes maintes fois lorsqu'elles répondaient aux besoins en information tout en étant plus efficaces que les classiques fichiers de texte ou les tableurs.

14.3 RETROUVER L'INFORMATION EFFICACEMENT

Il est courant pour les organisations de se doter d'un système de gestion documentaire, que ce soit via une série de répertoires partagés sur le réseau interne ou via une application de gestion de contenu. Typiquement, on y définit une structure documentaire plutôt complexe dans laquelle les personnes contribuant aux projets, aux applications et aux processus déposent les différents documents qu'elles créent ou modifient.

Tous les intervenants se font un devoir de déposer les documents dans cette structure mais pas nécessairement à la bonne place et en n'appliquant pas une nomenclature appropriée. Au bout de quelques mois, ce dépôt de documentation devient un bourbier dans lequel il devient difficile de retrouver l'information. Cela occasionne une perte de temps précieux à chercher plutôt qu'à trouver l'information. Selon notre expérience, cette situation est fréquente et sa probabilité augmente avec la taille de l'organisation.

Pour éviter cette situation indésirable, quelques règles simples doivent être appliquées :

1. Utiliser une **nomenclature** éloquente de fichiers, de répertoires ou de sections de sorte que quiconque sache ce qu'ils contiennent juste en voyant leur nom, sans avoir à les ouvrir. La définition de cette nomenclature doit être facilement

1. Par exemple, javadoc pour Java et nDoc pour l'environnement .Net de Microsoft.

accessible dans les actifs des processus et communiquée à tous à chaque fois qu'elle évolue.

2. Limiter la **classification** de l'information à trois catégories principales :

a) **Projet** : toute l'information relative à un projet et dont la durée de vie utile est liée au projet, tel que plans, calendriers, rapports de suivi des risques et problèmes, rapports sur l'état d'avancement, mesures et indicateurs.

b) **Application** : toute l'information relative à l'application et dont la durée de vie utile est liée à cette application, tel que spécifications, données de conception, architecture, code source, plans et résultats de test pour une version donnée, recette de construction d'un exécutable, et guide d'utilisation.

c) **Processus** : toute l'information relative au processus et dont la durée de vie utile est liée à ce processus, tel que flux de travail, glossaire et définitions, gabarits, guides, matériel de formation, mesures de performance et tout autre actif de processus jugé utile et nécessaire.

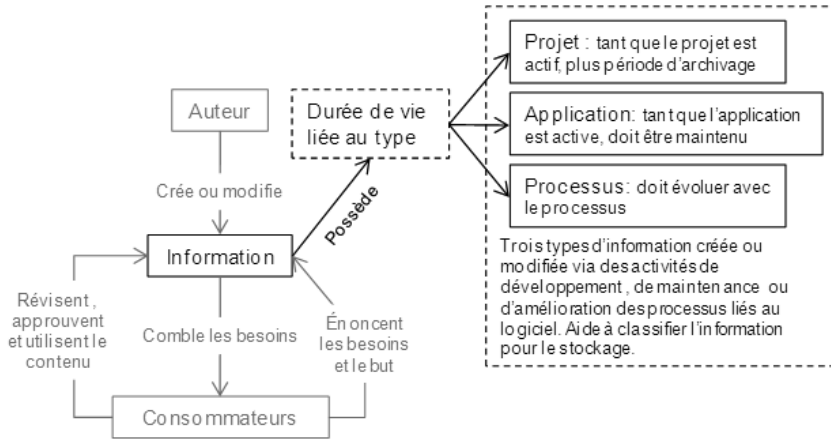


Figure 14.2 — Les trois types d'information créés ou modifiés via les activités de développement.

1. **Auditer** la structure régulièrement pour qu'elle reste claire et ainsi s'assurer de retrouver l'information efficacement. Cela signifie qu'il faille parfois renommer un fichier ou le déplacer pour qu'il soit à sa place. Les personnes les plus compétentes pour effectuer cet audit sont respectivement les chargés de projet, les responsables des produits et les propriétaires des processus.

Retour d'expérience : Une structure documentaire désorganisée

Nous avons pu constater à maintes reprises à quel point la structure documentaire devenait désorganisée après seulement quelques mois d'utilisation. Les conséquences sont des pertes de temps significatives pour retrouver l'information. Devant la difficulté à retrouver l'information cherchée, certains membres d'équipe s'adressent directement à l'auteur connu d'un document, qui leur transmet alors une version locale à partir

de son poste de travail. La version transmise est souvent désynchronisée avec celle recherchée sans que l'utilisateur ne s'en rende compte.
Cette situation est un symptôme à l'effet que la structure doive être réorganisée.
Sylvie et Mathieu

Il ne s'agit pas ici de forcer un outil unique car chaque équipe, projet ou organisation a besoin de flexibilité pour répondre aux besoins en information. Il est d'ailleurs fréquent de trouver trois outils : un gestionnaire de fichiers avec structure de répertoires, un outil de gestion du code et un intranet informationnel sur les processus.

14.4 EXEMPLES DE DOCUMENTATION DANS DES CONTEXTES SPÉCIFIQUES

Dans la présente section, nous vous proposons d'analyser le niveau de documentation des contextes que nous rencontrons fréquemment lors de nos interventions. Bien que ce ne soit que des exemples et que le contexte de votre organisation peut être différent, ils illustrent la différence d'analyse selon des contextes particuliers.

14.4.1 La modélisation

Valeur agile : Les individus et les interactions davantage que les processus et les outils.

La révision des documents de modélisation fait souvent l'objet de discussions animées. Principalement parce que ces documents opposent deux réalités : l'activité de conception et les exigences de la pérennité.

Dans le contexte d'une équipe *XP*[1], les **consommateurs** des documents sont les spécialistes de la modélisation[2] et leur **but** est la conception rapide et facile d'une solution logicielle. Le **profil** de ces consommateurs ne requiert pas un formalisme documentaire précis car les ateliers sont tenus en groupe et la proximité de ce groupe augmente la compréhension et les échanges entre les concepteurs. Ce qui leur est **utile,** c'est une documentation facilement modifiable, permettant l'erreur et favorisant les échanges.

C'est pourquoi ils **utilisent** des tableaux blancs et capturent et communiquent le résultat de leurs travaux avec des photos publiées sur des pages wikis. Les photos sont **utilisables**, car elles sont évocatrices et permettent de garder en mémoire les décisions prises au moment de la conception. Pour eux, **le temps d'attente** provoqués par le transfert de la conception dans un outil de modélisation formel et **les traitements superflus** pour respecter les formalités de l'outil sont des formes de **gaspillage**.

1. XP (*eXtreme Programming*) : méthode agile technique décrite au chapitre *2 Les méthodes agiles*.
2. Tous les spécialistes de la modélisation logicielle, notamment les développeurs, les architecteurs, les analystes d'affaire et TI, les pilotes, etc.

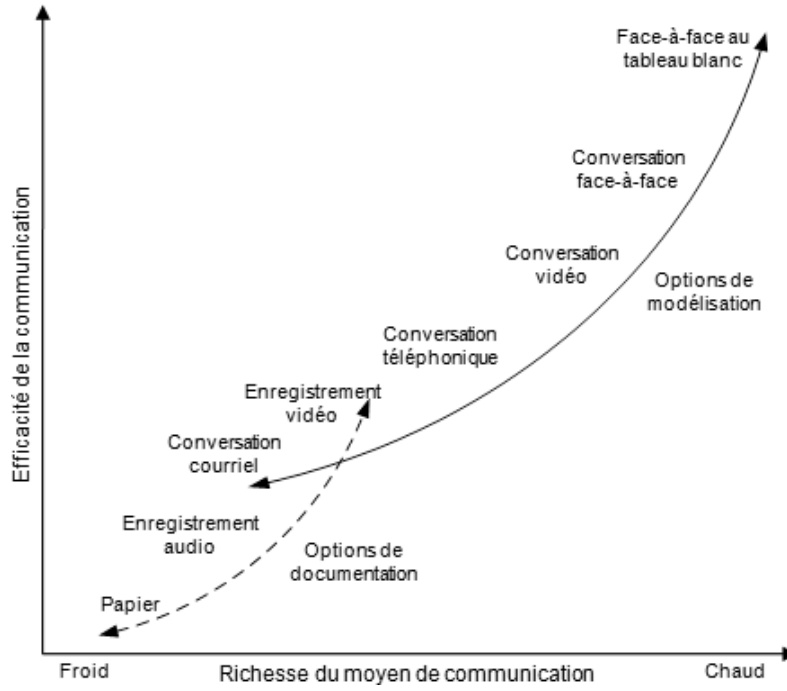


Figure 14.3 — Lorsqu'elle est possible, la conversation en face à face est toujours plus efficace que l'échange de document[a].

a. Traduit d'Allistair Cockburn, *Agile Software Development,* Addison-Wesley, 2001.

Principe agile : La méthode la plus efficace pour transmettre l'information est une conversation en face à face.

Le même contenu, dans un contexte de pérennité à l'intention d'une équipe d'entretien, ne rencontre pas les mêmes objectifs. Les **consommateurs** sont de nouveaux arrivants dont leur **but** est de comprendre le fonctionnement du produit logiciel. Ce qui est **utile** pour eux, c'est une documentation décrivant ce qu'il n'est pas possible de déduire à partir du code et du produit exécuté, comme les raisons expliquant les choix de conception. Parce que les concepteurs ne sont plus toujours présents, les documents ont avantage à être normalisés afin de permettre à n'importe quel intervenant ayant un **profil** de mainteneur de comprendre de manière autonome. Pour être **utilisable**, l'information doit être structurée de façon à ce que la recherche et le référencement soient efficaces au moment de corriger d'urgence un système. Pour eux, les **documents incomplets et manquants** et leur **indisponibilité** sont des formes de **gaspillage**.

Comme stratégie pour délier cette impasse, il faut reconnaître que les documents de conception ont un cycle de vie équivalent à la durée de vie du produit. Avant une itération, il n'est pas nécessaire de produire une documentation formelle, puisque les concepteurs n'en ont pas besoin. Cependant, les travaux de documentation destinée à la pérennité peuvent être inclus dans les activités des itérations, assurant qu'à la fin

de chaque itération, le dernier incrément de logiciel est prêt pour la production, et que sa pérennité soit ainsi assurée.

14.4.2 La conformité à des normes, lois ou règlements

> **Principe agile :** La simplicité - l'art de maximiser la quantité de travail à ne pas faire - est essentielle.

Il arrive parfois qu'une documentation soit coûteuse à produire, peu utilisée, mais demeure nécessaire. C'est particulièrement vrai lorsqu'une solution doit se conformer à des normes et fait l'objet d'audit.

Retour d'expérience

Je suis intervenu auprès d'une entreprise qui développait des outils pour concevoir des plans de vol à l'intention des pilotes d'avion ayant besoin d'un corridor sécuritaire. Leur produit logiciel était capable de considérer tous les obstacles et risques possibles autour de la piste d'un aéroport. Ce produit était innovant car il était construit autour d'une nouvelle méthode de calcul se voulant plus fiable et juste que celle utilisée par ses concurrents.

Pour être homologué, le produit avait fait l'objet d'un audit par les autorités de transport aérien. L'une des conclusions était une documentation insuffisante. Le créateur jugeait l'audit sévère et injuste, puisque le produit répondait aux normes en matière de plan de vol et que la norme suffisait pour en évaluer la conformité. Malheureusement, il fallait donner raison à l'auditeur sur l'aspect de la documentation.

Les normes existantes ne documentent pas la solution logicielle, elles servent plutôt de cahiers de charges indiquant tout ce que le logiciel DOIT être en mesure de faire. Cependant, elles n'indiquent pas COMMENT le logiciel répond aux exigences de l'industrie. Bien que cette documentation soit difficile à produire, elle est aussi primordiale que les fonctionnalités et elle doit être incluse dans chacun des incréments de logiciel du projet. C'est une bonne pratique, dans ce cas, d'inclure la documentation de conformité aux normes dans la définition de terminé.

Mathieu

Dans le contexte de la conformité, les **consommateurs** sont les auditeurs et leur **but** est de vérifier qu'un produit ou un processus répond bien aux normes. Le **profil** d'un auditeur est celui d'un expert et ce qui lui est **utile** est de comprendre et mesurer comment le produit logiciel répond et applique les normes. Bien qu'il ne la consulte pas fréquemment (utilisation), la documentation doit décrire de façon exhaustive comment la norme est appliquée, en termes de couverture à travers le temps. Elle est **utilisable** lorsqu'elle permet cette comparaison claire entre le logiciel et la norme, tout en utilisant les termes précis de la norme.

Les formes de **gaspillage** d'une documentation traitant de la conformité sont les suivantes :

- une documentation **incomplète** et manquante ;
- une documentation reprenant le texte de la norme, infligeant un **traitement superflu** des processus aux travaux de rédaction ;
- une documentation qui n'est pas mise à jour régulièrement, augmentant les chances qu'elle comporte des **défauts** par manque de rectitude ou manque d'historique.

En résumé

La documentation est un élément important à considérer en support à l'incrément de logiciel. Contrairement à la croyance populaire, les approches agiles lui accordent cette importance. Cependant, le *Manifeste agile* incite à être critique envers le processus de documentation d'une organisation, qui doit se concentrer sur la production de documents utiles plutôt que sur la conformité d'un processus.

En ce qui concerne la documentation générée par les projets agiles, les équipes, via les rétrospectives, doivent s'assurer d'être efficaces. La règle des trois « U » (Utile, Utilisable et Utilisé), l'identification des consommateurs et la réduction de gaspillage sont des pistes de solutions permettant d'améliorer la qualité de la documentation et d'augmenter le rendement de l'investissement des activités de rédaction.

15

Mesures de performance

Objectif

Dans ce chapitre, vous découvrirez l'essentiel des mesures et indicateurs dont l'utilisation est essentielle pour améliorer la performance des processus. Ils peuvent être mis en œuvre parallèlement aux pratiques agiles. Mettre en place des mesures de performance peut avoir certains effets pervers et vous verrez comment établir un noyau de mesures objectives qui conviennent à l'organisation et aux équipes.

Mise en situation : Jackie et la performance de son organisation

Jackie est vice-présidente des technologies de l'information (TI) dans une grande société financière. Relevant du CIO, elle vient d'arriver en poste et il lui a demandé de faire état de la situation.

Jackie a pu mettre la main sur la majorité des informations dont elle a besoin pour analyser la situation à partir des dossiers de gouvernance des projets TI contenant les mesures et indicateurs des douze derniers mois[1] :

Description	Valeur
Nombre de systèmes	168
Dépenses en sous-traitance de projets de développement et d'entretien	54 millions §

1. § : symbole générique de monnaie.

Dépenses en sous-traitance de gestion de l'infrastructure	22 millions §
Nombre de projets de développement et d'entretien	67
Taille des projets (en jours-personnes)	52 à 4470 [moy.= 860]
Durée moyenne des projets	11 mois
Dépassement budgétaire moyen	5 %
Dépassement moyen de l'échéancier	1 %
Utilisation moyenne de la contingence budgétée	De 25 % à 100 %
Nombre moyen de défauts post-déploiement	6,2
Coût unitaire moyen	Inconnu
Taille moyenne des logiciels par projet	Inconnue

En regardant ces chiffres, Jackie comprend pourquoi son patron veut réduire les dépenses en TI de façon significative. Le moindre petit projet d'évolution des systèmes coûte un demi-million **§**. Les délais entre le démarrage et la livraison semblent s'allonger depuis plusieurs années, à un point tel que la société a rendu disponibles des fonctionnalités en ligne à ses clients plus tard que ses compétiteurs, ce qui lui a coûté une part de marché. La situation est inacceptable dans un contexte où les TI sont critiques aux affaires et à la survie de l'organisation.

Il subsiste des inconnues importantes que Jackie va s'employer à découvrir au cours des prochains mois. Elle a besoin d'en savoir davantage sur le coût unitaire et souhaite faire une analyse d'étalonnage avec l'industrie financière. Jackie a aussi besoin de connaître le niveau de couplage inter-systèmes. Elle le soupçonne d'être plutôt élevé étant donné le symptôme d'allongement des délais de livraison des projets.

Pour l'éclairer, Jackie confie la réalisation des deux mandats à des experts externes, puisqu'elle sait pertinemment que les expertises d'étalonnage et de diagnostic d'architecture ne sont pas disponibles à la banque. Les experts ont besoin de 2 à 3 mois pour réaliser leurs travaux.

(À suivre...)

15.1 CONCEPTS CLÉS DE LA MESURE

Retour d'expérience : une avalanche de mesures et d'indicateurs

Je suis intervenue pour déployer la méthode Scrum dans une organisation développant un produit logiciel dans le domaine de la sécurité physique de lieux publics. Le président était enthousiaste à me montrer le vaste ensemble de données mesurées.

Lorsque je lui ai demandé quelles étaient les analyses de données qui en découlaient et quelles décisions il fondait sur ces analyses, sa réponse a été « aucune ». Autrement dit, il dépensait de l'argent à cumuler des données inutilement, ce qui est considéré comme du gaspillage. Nous avons donc travaillé son programme de mesure afin de le rendre utile et nécessaire en appliquant les principes essentiels de la mesure.
Sylvie

Il nous arrive régulièrement d'intervenir dans des organisations maîtrisant peu les concepts de base de la mesure et de la métrologie[1]. Cette conclusion, nous la tirons en observant un certain nombre de symptômes :

- **Peu ou pas de mesures** : aucun programme de mesures en place, les décisions sont fondées sur des intuitions plutôt que sur des données fiables et cohérentes avec les besoins en information.
- **Trop de mesures** : des mesures sont cumulées sans être analysées ni utilisées, ce qui provoque un gaspillage d'effort.
- **Mauvaises mesures** : des mesures sont utilisées à des fins autres que celles pour lesquelles elles sont destinées, causant des effets pervers indésirables. Par exemple, si un gestionnaire utilise le nombre de lignes de code produite par développeur par année pour attribuer les augmentations salariales et bonus, il obtiendra des programmes de taille gonflée inutilement et diminuant leur maintenabilité, sans parler de la compétition malsaine qui s'installera dans ses troupes.
- **Résultats de mauvaise qualité** : les données de base composant les indicateurs sont erronées ou incomplètes. Par exemple, lorsque tous intervenants d'un projet indiquent avoir travaillé 35 heures par semaine, soit le nombre d'heures pour lesquelles ils sont payés, alors qu'ils ont tous travaillé 50 heures par semaine en moyenne.

Retour d'expérience : Quand les données mentent...

Il m'est arrivé à plusieurs reprises d'intervenir auprès de sociétés dont les données d'effort étaient de mauvaise qualité. Le cas typique se produit dans des cultures de contrôle où la responsabilité du respect des budgets est attribuée aux chargés de projet et dont les paramètres de budget, de date de livraison et de contenu sont fixés par la direction, avec peu de marge de manœuvre.

Dans ce cas, les chargés de projets ferment le code de projet au moment où les coûts atteignent le budget et la suite des travaux se poursuit sous un autre code de projet ou sous le budget d'entretien, et avec l'accord des gestionnaires. Il devient alors difficile, voire impossible, de reconstituer l'effort réel pour livrer l'ensemble des travaux.

C'est ce que nous appelons « se mentir à soi-même » puisque l'organisation fausse l'effort réel à partir duquel est calculé le coût de projet. Le cas se corse quand ces organisations utilisent leurs données historiques de projet pour budgéter les prochains projets, où non seulement elles se sont menti, mais elles croient à leurs mensonges. C'est

1. Métrologie : science de la mesure.

ce que nous appelons un « cas pathologique » découlant d'un problème systémique qui doit d'abord être résolu pour que les comportements changent.
Sylvie

15.1.1 Processus de mesure

Dans cette section, nous survolons le processus de mesure en génie logiciel, soit la norme ISO 15939, mise en œuvre par la méthode PSM – *Practical Software measurement*. Dans un contexte agile, nous verrons comment nous pouvons appliquer cette norme afin de mesurer les améliorations apportées aux façons de faire.

La norme ISO 15939

Nous apprécions le caractère itératif et incrémental de cette norme qui en fait un outil idéal pour établir ou améliorer le programme organisationnel des mesures lors d'une transition agile, particulièrement lorsque la stratégie d'adoption « en sandwich »[1] est appliquée. Le processus de mesure proposé dans cette norme comporte quatre étapes (Figure 15.1).

- **Établir et soutenir l'engagement envers la mesure** : à la base, le processus de gestion de l'organisation doit avoir suffisamment de maturité pour que les dirigeants s'engagent à établir et maintenir des activités de mesure. Cela va de soi dans une transition agile quand les organisations traitent leurs problèmes ou opportunités avec des stratégies planifiées et des objectifs clairs. L'amélioration des processus ne saurait se faire sans mesure, c'est un principe élémentaire.
- **Planifier le processus de mesure** : cette étape consiste à établir le plan de mesure décrivant les aspects typiques du plan (pourquoi, quoi, qui, quand, comment, combien) et s'assurant des relations entre :
  - Les **objectifs d'affaires** quantifiables : certains objectifs peuvent alimenter le choix d'une stratégie tandis que d'autres objectifs découlent du choix d'une stratégie.
  - Les **besoins en information** : quelles informations sont requises au moment de faire une transition agile ? Typiquement, celles permettant de vérifier que les objectifs recherchés sont atteints ou en voie de l'être. Elles permettent également de vérifier certaines hypothèses nécessitant d'être vérifiées.
  - Les **indicateurs** : obtenus suite à l'analyse impliquant plusieurs mesures de base ou dérivées, comme le coût unitaire moyen d'un ensemble de projets de même technologie complétés dans une période donnée ou la densité des défauts des douze derniers mois. Il est important de définir des indicateurs répondant aux besoins en information des consommateurs qui en feront l'usage, c'est-à-dire les décisions qu'ils prendront basées sur les résultats de l'indicateur. Pour assurer la cohérence dans l'usage des indicateurs,

1. Se référer au chapitre *11 Gouvernance*, dans la section sur les stratégies d'adoption.

l'interprétation des résultats doit être connue ou définie dans le contexte de leur utilisation.

– Les **mesures dérivées** : obtenues en appliquant une formule de calcul utilisant plusieurs mesures de base, comme un coût unitaire (heures par points de fonction) ou la vélocité (nombre de points d'effort terminés dans une itération). Pour assurer sa qualité, il est nécessaire d'indiquer la formule de calcul menant aux résultats.

– Les **mesures de base** : obtenues en appliquant une méthode de mesure quantifiant un attribut observable d'un projet ou d'un logiciel dans une unité de mesure définie. Elles sont des éléments qui se comptent, comme des heures, des défauts, des points de fonction et des jours écoulés. Ces mesures sont définies dans un plan de mesure : unité de mesure, précision requise, qui fait la mesure, procédure de collecte des données, source des données, où sont stockées les données et les mécanismes d'assurance qualité.

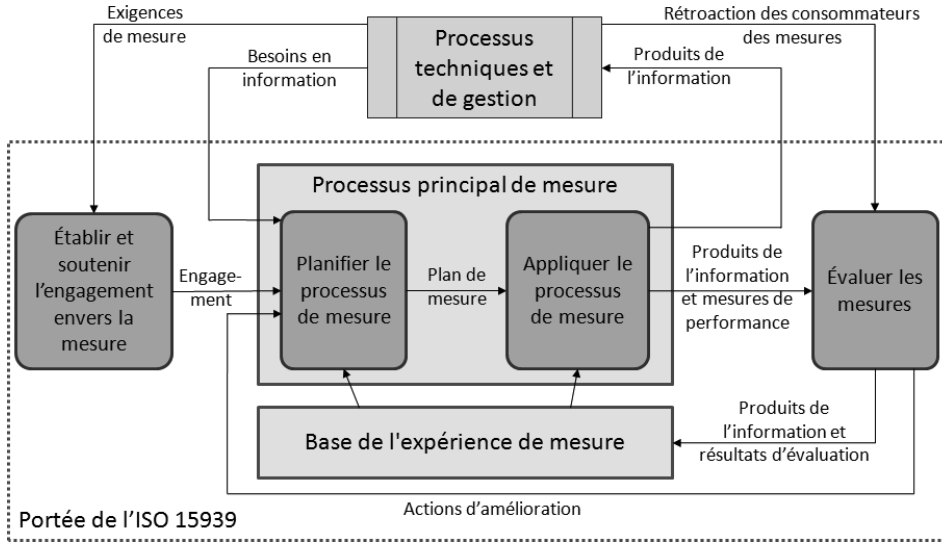


Figure 15.1 — Aperçu de la norme ISO 15939.

- **Appliquer le processus de mesure** : c'est mettre le plan de mesure à exécution et fournir les résultats fiables et intègres à leurs consommateurs. Ces résultats sont appelés « produits de l'information » dans la .
- **Évaluer les mesures** : régulièrement, il faut revoir si les données collectées servent à alimenter les décisions pour leur adéquation aux objectifs changeants, leur capacité à combler les besoins en information et la possibilité d'en améliorer une caractéristique, tel que rendre leur procédure de collecte plus efficace.

Exemple de mise en œuvre de l'ISO 15939 dans un contexte agile

Dans cette section, nous présentons un cas typique de l'application de la norme ISO 15939.

Tout d'abord, un engagement envers la mesure est nécessaire afin de s'assurer du soutien de la direction. Sinon, l'activité risque d'être sous-financée et les résultats pourraient être incomplets et de mauvaise qualité. Pour donner cet engagement, les dirigeants doivent être convaincus que les informations dont ils disposent sont insuffisantes pour assurer la qualité des décisions qu'ils pourraient être amenés à prendre pendant la transition agile.

Puis, on commence à établir un plan de mesure en collectant les objectifs et les besoins en information qui en découlent. Un plan de mesure peut sembler complexe et long, mais nous avons l'habitude de le faire tenir sur trois onglets dans un tableur, que ce soit pour une petite (moins de 50 personnes en développement) ou une grande (plus de 200 personnes en développement) organisation. Par principe, nous préférons commencer par une forme simple et de la complexifier seulement si la forme simple ne convient plus aux besoins.

Dans le premier onglet, nous inscrivons les objectifs mesurables (Tableau 15.1).

Tableau 15.1 — Exemple d'objectif mesurable.

#	Objectif	Responsable
1	Réduire les délais de livraison d'au moins 20 %	Comité de gestion

Dans le deuxième onglet, nous inscrivons les besoins en information, sous forme de questions menant à des indicateurs spécifiques (Tableau 15.2).

Dans le troisième onglet, nous inscrivons les mesures de base et les mesures dérivées nécessaires pour alimenter les indicateurs et les besoins en information (Tableau 15.3).

Cet exemple représente un sous-ensemble d'un plan de mesure ne nécessitant pas d'outils complexes.

Conseil : ne cherchez pas à compliquer votre plan de mesure inutilement. Démarrez-le simplement et améliorez-le avec le temps. Pour commencer, mettez-y les mesures et indicateurs que vous utilisez déjà pour vos activités de gestion de projets et de gouvernance.

Tableau 15.2 — Exemples de besoin en information et des indicateurs qui en découlent.

Besoins en information	Quelle est la réduction relative des délais de livraison avant et après l'adoption de l'agilité ?
Liés à l'objectif no.	1- Réduire les délais de livraison d'au moins 20 %
Indicateurs	Réduction relative des délais de livraison
Source des résultats	Base de données des projets du bureau de projet
Présentation des résultats	Par secteur d'activités, « réduction relative des délais » sur le rapport consolidé des performances des projets
Formule de calcul	(((délai moyen de livraison)$_{trad}$ – (délai moyen de livraison)agile) / (délai moyen de livraison)$_{trad}$) * 100
Fréquence	Trimestrielle
Responsable	Bureau de projet
Consommateurs	Comité de gestion
Procédure d'analyse[a]	Surligner en vert si l'écart >= 20% Surligner en jaune si l'écart est entre 0 % et 20 % Surligner en rouge si l'écart est négatif
Actions potentielles	Écart en deçà de l'objectif : le bureau de projet doit investiguer les raisons et recommander des actions.

a. Cette procédure d'analyse est simple, mais elles ne le sont pas toutes. Dans certains cas plus complexes, il peut être utile d'insérer un petit document explicatif dans la case ou un hyperlien vers l'information sur l'intranet corporatif.

Tableau 15.3 — Exemples de mesures alimentant les besoins en information.

Mesure	(Délai moyen de livraison)$_{trad}$	Délai de livraison d'un projet
Unité de mesure	Jours ouvrables	Jours ouvrables
Type de mesure	Dérivée	De base
Précision requise	1 jour	1 jour
Qui mesure ?	Bureau de projet	Chargé de projet
Source des données	Base de données des projets du bureau de projet	Bilan de projet
Quand mesurer ?	Référentiel au 1er janvier 2011	À chaque livraison
Où stocker les résultats ?	Rapport consolidé des performances des projets, après le 1er janvier 2011.	Base de données des projets du bureau de projet
Procédure de collecte	Calculer la moyenne des délais de livraison de l'ensemble des projets non agiles, complétés depuis le 1er janvier 2009.	Compter ou calculer le nombre de jours ouvrables entre la date de démarrage de la version et la date de livraison de cette version (ne pas tenir compte des week-ends et des jours fériés).
Procédure d'assurance qualité	Le bureau de projet passe en revue l'ensemble des projets non agiles complétés au cours de la période visée et s'assure que la liste est complète. Il corrobore chaque délai de livraison avec chacun des bilans des projets ciblés.	Le bureau de projet refait le calcul. S'il y a un écart, vérifier avec le chargé de projet et inscrire la valeur vérifiée.

15.1.2 Principes essentiels

Après plus de quinze années d'expérience à mettre en place des programmes de mesure dans les organisations, en soutien à leurs démarches d'amélioration, nous en sommes venus à appliquer les principes suivants :

1. **Mesurer le processus, et non pas les individus** : les mesures ciblant la performance des individus créent souvent plus de mal de que de bien en tuant l'esprit d'équipe, un élément essentiel en agilité. Les mesures ciblant le processus sont en général accueillies favorablement par l'ensemble du personnel lorsqu'il prend le parti de l'amélioration.
2. **Les relations** entre les mesures, les indicateurs, les objectifs d'affaires et les stratégies **doivent être clairement établies** : sinon, l'organisation gaspille des ressources à collecter et analyser des données erronées ou périmées.
3. **Assurer la viabilité économique** d'une mesure : au moment d'évaluer une mesure, il faut s'assurer que le coût global de la mesure demeure inférieur à la valeur d'affaires découlant de l'usage qui en est fait. Toutefois, il ne faut pas négliger le coût associé au retrait de cette mesure. La conséquence du retrait est le retour à des décisions basées sur des impressions plutôt que des faits.
4. Définir les mécanismes de collecte, de validation et d'analyse des données **en évitant les sept formes de gaspillage**[1].
5. **Donner de la rétro-information** (*feedback*) à ceux qui collectent les données : notamment en rendant les résultats transparents faisant en sorte que les personnes collectant les données auront naturellement tendance à être plus précises lors de leur collecte. Le cas classique est celui de l'enregistrement des heures travaillées dans le système de feuilles de temps où, lorsque les personnes comprennent l'utilité des données, saisissent plus précisément ces heures.
6. **Assurer la qualité des données mesurées** : une donnée de mauvaise qualité rendra les résultats des indicateurs non crédibles, allant jusqu'à engendrer de mauvaises décisions.

> Adage entendu : « *La qualité d'une décision dépend de la qualité des données ayant servi à l'alimenter* ».

15.2 MESURER LA PRODUCTIVITÉ DU PROCESSUS

Plusieurs gestionnaires des TI expriment leur besoin d'améliorer la productivité de leurs projets et de leur processus de développement logiciel. Lorsqu'ils le font, cette mesure est plus souvent qualitative que quantitative.

1. Les sept formes de gaspillage, discutées au chapitre 8 *Infrastructure technologique*, sont : travaux incomplets, caractéristiques ou fonctionnalités superflues, réapprentissage, permutation entre les tâches, transfert de tâches ou de composants, délais et défauts et corrections.

Les trois indicateurs de gestion de projet, illustrés aux cimes du triangle de la Figure 15.2, sont en général bien maîtrisés par les organisations. La pratique de la gestion de projet logiciel avec ces indicateurs suggère que si toutes les fonctionnalités sont livrées avec niveau de qualité prévue, à la date prévue et à l'intérieur du budget prévu, alors l'équipe et le chargé de projet ont été efficaces.

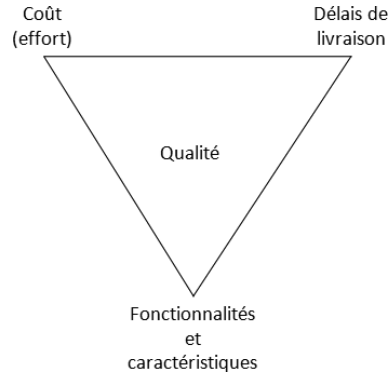


Figure 15.2 — Les indicateurs classiques en gestion de projet logiciel.

En réalité, ils n'ont été efficaces qu'à estimer, pas nécessairement efficaces à livrer le projet TI. De plus, les statistiques du *Standish Group* démontrent que seulement 16 % à 32 % des projets TI parviennent à livrer à l'intérieur des budgets consentis[1].

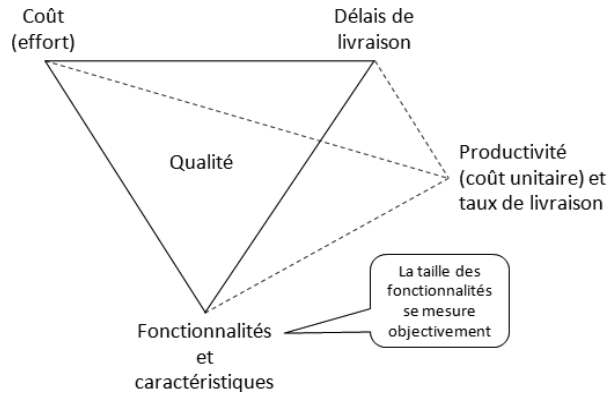


Figure 15.3 — La mesure de productivité met en correspondance la taille des fonctionnalités avec le coût (effort) et la mesure du taux de livraison met en correspondance la taille et les délais de livraison.

Pour en avoir une idée plus précise, ces indicateurs doivent être mis en correspondance pour mesurer la productivité réelle. Avec une mesure normalisée, il est possible de comparer les projets entre eux, en plus de comparer la productivité des projets avec d'autres projets ailleurs dans le monde (étalonnage ou *benchmarking*).

1. The Standish Group, *Chaos Report*, de 1994 à 2009, http://www1.standishgroup.com .

15.2.1 Pourquoi mesurer la productivité du processus de développement ?

Les indicateurs classiques de gestion de projet sont incomplets pour permettre une amélioration quantifiée de la productivité. Il faut leur ajouter la mesure de la taille fonctionnelle des projets pour y arriver. Notre expérience et d'autres études[1] ont démontré une forte corrélation entre la taille fonctionnelle et l'effort réel d'un projet, ainsi qu'une forte corrélation entre la taille fonctionnelle et les délais de livraison (Figure 15.4).

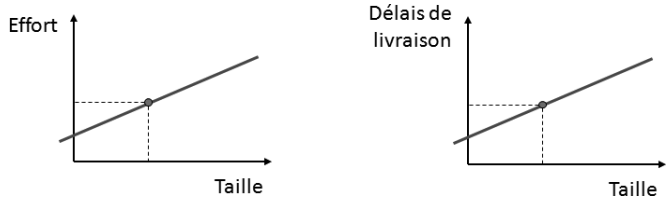


Figure 15.4 — Il existe une forte corrélation entre la taille fonctionnelle et l'effort réel, ainsi qu'entre la taille fonctionnelle et les délais de livraison.

Donc, une organisation ayant mesuré la taille fonctionnelle de ses projets est capable d'établir des modèles de productivité et d'estimation fiables quand ses données d'effort et de délais sont de qualité et que les facteurs de corrélation sont élevés. Notre expérience démontre que la productivité ainsi mesurée représente non pas la productivité d'équipes ou d'individus mais bien celle du processus car, pour les organisations appliquant le même processus à travers l'ensemble des projets mesurés, les résultats sont comparables alors que les projets sont réalisés par des équipes différentes.

La taille fonctionnelle, un ingrédient clé

Typiquement, la mesure de la taille fonctionnelle des logiciels répond à des besoins liés aux activités suivantes :

- **Productivité** : pour la connaître et ainsi la contrôler et l'améliorer.
- **Estimation** : parce qu'une productivité connue et stable permet d'améliorer les estimations précoces des projets, ce qui est reconnu comme un point faible de la plupart des méthodes agiles.
- **Étalonnage (aussi appelé « *benchmarking* »)** : pour comparer la performance de son processus de développement avec celle d'autres unités TI de l'organisation ou externes, ailleurs dans le monde.
- **Gouvernance** : notamment pour contrôler les coûts relatifs d'entretien d'un portfolio de logiciels et prendre action quand les résultats débordent des valeurs attendues pour certains d'entre eux.

Pour combler l'un ou l'autre de ces besoins, la taille fonctionnelle doit permettre :

1. Source :Abran, A., *Software Metrics and Software Metrology*, Wiley, 2010.

- De l'**objectivité** : une méthode normalisée appuie la mesure réduisant les risques d'interprétation subjective par les personnes qui mesurent ;
- De la **répétitivité** : une personne mesure le même projet deux fois à des moments différents, en se servant des mêmes informations, obtiendra le même résultat ;
- De la **reproductibilité** : deux personnes différentes font la mesure séparément et obtiennent des résultats similaires, voire identiques.

Aussi, cette taille doit être indépendante de la technologie et de la méthodologie de développement. Il est important ici de distinguer la taille du logiciel du coût de développement : ce sont deux données complètement différentes.

15.2.2 Comment mesurer la productivité et autres indicateurs similaires ?

La mesure de productivité s'exprime de deux façons, en combinant l'effort en heures et la taille fonctionnelle en points de fonction :

- **Ratio de productivité du processus** : en points de fonction par mois-personne (mesure d'effort), bien que la définition d'un « mois-personne » puisse être variable d'une organisation à l'autre : par exemple est-ce quatre semaines de 35 heures [= 140 heures] ou 4,3 semaines de 40 heures [= 172 heures] ?
- **Ratio de coût unitaire** : en heures par point de fonction.

En combinant la taille fonctionnelle avec les délais de livraison, on obtient un ratio de taux de livraison, exprimé en points de fonction par mois. Ce peut être une moyenne par mois ou un profil réparti sur une année type afin de tenir compte des périodes creuses comme celle des vacances ou des congés fériés regroupés.

Les différentes méthodes de mesures de la taille fonctionnelle

Au début des années 2000, l'organisation internationale de normalisation (ISO) a publié un cadre normatif, la norme ISO 14143, définissant ce qu'est une mesure de taille fonctionnelle. Au moment d'écrire ce chapitre, il existait cinq méthodes de mesure de la taille fonctionnelle conformes au cadre normatif ISO :

- IFPUG v4.1 (*International Function Point Users Group*), norme ISO 20926.
- NESMA v2.1 (*Netherlands Software Measurement Association*), norme ISO 24570
- FiSMA (*Finland Software Measurement Association*), norme ISO 29881.
- Mark II, norme ISO 20968.
- COSMIC v3 (*Common Software Measurement International Consortium*), norme ISO 19761.

La méthode COSMIC, la seule dite de « deuxième génération », a été développée au Canada et en Europe, fruit d'une collaboration internationale. Ses concepteurs ont veillé à ce que les principes de métrologie soient respectés, rendant les résultats issus de la méthode plus objectifs, reproductibles et répétables que les méthodes de première génération. En plus des systèmes d'information, la méthode COSMIC

s'applique à plusieurs autres types de logiciels : services Web et services d'affaires, systèmes embarqués[1], systèmes temps réel et entrepôts de données.

Pourquoi la méthode COSMIC ?

De nos jours, les sociétés développent un large éventail de types d'applications sous diverses plateformes. Les limites des méthodes de première génération ont mené plusieurs organisations à adopter COSMIC principalement parce qu'elle permet de mesurer la majorité de leurs applications, qu'elles soient client-serveur, Web, composantes de services d'affaires, entrepôts de données ou logiciels embarqués. C'est aussi pour cette raison que le Japon a adopté COSMIC comme norme nationale de la mesure de la taille fonctionnelle du logiciel en 2003, peu après sa publication en tant que norme ISO 19761.

Étant donné que COSMIC applique les concepts et principes de la métrologie, les résultats obtenus sont plus précis, tant les petites fonctionnalités que les très grandes qu'avec les autres méthodes. Cette caractéristique importante favorise des données de meilleure qualité menant à des modèles d'estimation plus fiables.

Pour pouvoir mesurer la taille avec les autres méthodes, les mesureurs doivent connaître les attributs des données manipulées par les fonctionnalités, soit quand l'analyse détaillée et la conception des données sont terminées. Avec COSMIC, ces détails sont superflus et une mesure peut-être faite dès que les objets de données d'affaires sont connus, sans connaître leurs attributs, soit relativement au début de l'analyse détaillée. Pour un usage d'estimation, c'est un avantage considérable puisque les résultats sont obtenus plus tôt dans le cycle de vie.

Quant aux organisations intéressées par l'étalonnage, il existe un plus grand nombre de données de projets mesurées avec IFPUG qu'avec n'importe quelle autre méthode de mesure. Cependant, des équations de conversion permettent de transformer les résultats IFPUG en points de fonction COSMIC, et la norme COSMIC est celle pour laquelle l'augmentation relative des données de projets soumises est la plus élevée[2].

Finalement, le guide de mesure de la méthode COSMIC est gratuit et disponible dans plusieurs langues, dont le français[3].

Pourquoi ne pas utiliser les points d'effort pour répondre à ces besoins ?

Une majorité de projets agiles mesure la taille de leurs exigences en points d'effort (*user story points*)[4]. Il est d'usage d'appliquer la suite de Fibonacci (0, 1, 2, 3, 5, 8, 13,

1. Un système est dit « embarqué » lorsque le logiciel est « brûlé » sur une puce insérée sur une carte électronique. Par exemple, des microprocesseurs dans une voiture, le logiciel d'un four à micro-ondes, des équipements médicaux et du matériel militaire (missiles, radar, système de commandement et contrôle) contenant du logiciel.
2. Source :*International Software Benchmarking Standards Group*, http://www.isbsg.org .
3. À télécharger du site http://www.cosmicon.com→*Portal*→*Download*→*Translations*.
4. Se référer au chapitre 6 *Gestion de projet* pour plus de détails au sujet des points d'effort.

21, etc.) pour attribuer une valeur combinant la taille et la complexité afin que cette valeur soit le reflet de l'effort requis pour réaliser l'item du carnet de produit. Ce sont donc des valeurs sur une échelle ordinale qui sont additionnées pour établir la taille d'une itération et, par extension, celle du projet.

L'attribution de la valeur est laissée aux membres d'une équipe qui s'entendent sur le nombre de points d'effort d'un item qu'ils considèrent de taille « moyenne ». Ensuite, tous les autres items sont comparés à l'item étalon pour attribuer leur valeur en points d'effort. D'un projet à l'autre, cette valeur attribuée à l'item étalon varie, immanquablement.

Les faiblesses de l'utilisation des points d'effort suivantes sont observables, au regard des besoins liés à la mesure de la taille du logiciel :

1. Il n'est pas possible de normaliser la valeur des points d'effort d'un projet à l'autre, ni d'une organisation à l'autre, car cette valeur est subjective et propre à l'équipe l'ayant attribuée.
2. D'un point de vue métrologique et mathématique, les opérations d'addition sur des valeurs d'une échelle ordinale ne sont pas admissibles. Ces valeurs ne peuvent être qu'ordonnées, ce qui enlève la crédibilité au total obtenu.
3. Les points d'effort représentent une estimation de l'effort, et non pas une mesure de la taille. Pour être objectif, répétable et reproductible, ces deux concepts, taille et effort, doivent pouvoir être séparés et être mesurés séparément, c'est-à-dire comptés dans leur unité de mesure. La taille et l'effort réel (en heures) se mesurent, mais les points d'effort s'estiment (dans le sens d'approximation, et sans en connaître les marges d'erreur).

Toutefois, nous ne recommandons pas nécessairement d'abandonner les points d'effort car ils ont leur utilité pour les membres d'équipe, notamment de permettre une discussion saine sur la quantité de travail requise telle que perçue par chacun au moment de la planification d'une itération. Cette estimation d'effort est nécessaire pour que les membres d'équipe prennent un engagement.

Comparativement à une taille normalisée, nous croyons que l'application des points d'effort sert à d'autres consommateurs et pour d'autres usages dans des contextes différents (Tableau 15.4). Ils peuvent alors être utilisés indépendamment.

Des travaux ont été publiés démontrant qu'il est possible d'appliquer une mesure de taille fonctionnelle sur les items d'un carnet de produit, à condition qu'ils représentent des fonctionnalités[1]. En effet, la taille fonctionnelle est nulle sur des items non fonctionnels du carnet de produit, comme la réingénierie d'une composante de code afin de permettre de futures évolutions planifiées. De plus, lorsqu'un coût unitaire moyen est établi, il s'applique à l'échelle des projets et non pas à chacune des fonctionnalités pour lesquelles la variation du coût unitaire peut être significative. À défaut d'une autre mesure servant aux membres d'équipe, les points d'effort demeurent utiles pour estimer l'effort des items.

1. Rule, Grant, *Sizing User Stories with* COSMIC, http://www.smsexemplar.com/author/grant-rule/, 2010.

Tableau 15.4 — Comparaison des paramètres d'utilisation de la taille fonctionnelle et des points d'effort.

Paramètre	Taille fonctionnelle avec COSMIC	Points d'effort
Consommateurs	Bureau de projet ou équipe de soutien au développement, haute direction	Membres d'équipe
Usage	1. Estimation préliminaire ou à haut niveau ; 2. Étalonnage ; 3. Quantifier l'amélioration de la productivité ; 4. Activités de gouvernance.	– Estimation d'effort par item d'un carnet de produit ; – Prendre un engagement en fonction de leur capacité.
Contexte	Gestion de la performance du processus et de l'organisation de développement logiciel.	Planification d'une itération ou d'une livraison.

15.2.3 La méthode COSMIC

Aperçu de la méthode COSMIC

Conceptuellement, que fait un logiciel ? Il déplace des données et il les transforme ou les manipule avant de les déplacer à nouveau. S'inspirant de ce concept, la méthode COSMIC mesure ces mouvements de données appliqués à chaque objet d'affaires, manipulé par chacune des fonctionnalités. Il existe quatre types de mouvement de données : des Entrées (E), des Sorties (S), des Lectures (L) et des éCritures (C) (Figure 15.5). Toute personne ayant travaillé un peu en développement logiciel maîtrise aisément ces concepts.

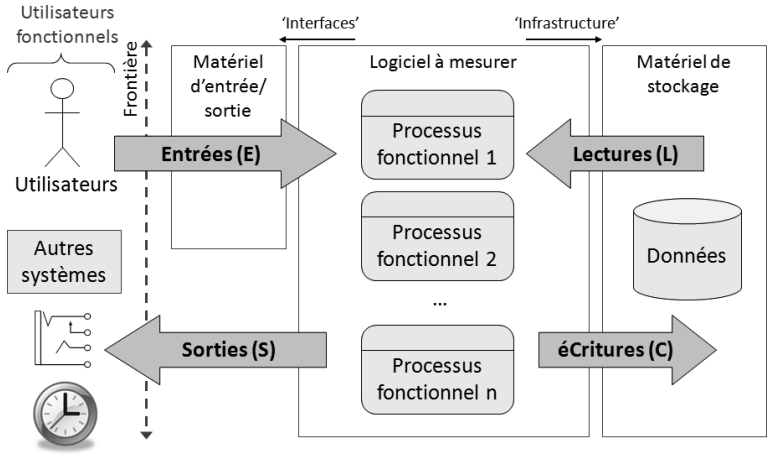


Figure 15.5 — Vue d'ensemble de la méthode COSMIC.

Cette vue d'ensemble de la méthode COSMIC illustre les besoins d'identifier éléments suivants :

- Les **utilisateurs fonctionnels** : dans la partie gauche de la vue d'ensemble, ils peuvent être des utilisateurs d'autres systèmes avec lesquels le logiciel à mesurer

s'interface, des dispositifs d'ingénierie (tels que valves ou capteurs) et l'horloge, parfois appelée « céduleur », responsable de démarrer les fonctionnalités s'exécutant en différé (*batch processing*).

- La **frontière** du logiciel à mesurer : de façon à bien identifier les fonctionnalités en faisant partie.
- Les **processus fonctionnels** : soit les fonctionnalités servant à l'un ou l'autre des utilisateurs fonctionnels identifiés. Un processus fonctionnel est systématiquement déclenché par une intervention de l'un des utilisateurs fonctionnels via un mouvement de données d'Entrée.
- Les **groupes de données** : soit les objets d'affaires manipulés par chacun des processus fonctionnels. Par exemple, ce peut être des « clients », des « adresses de clients », des « commandes », des « items commandés », des « taux de taxes » et tout autre objet d'intérêt pour les affaires. Chaque groupe de données contient au moins un attribut de données et il est unique et cohésif quant aux attributs de données qu'il contient.
- Les **mouvements de données** : les quatre types illustrés par des flèches pleines. Les mouvements d'Entrée et de Sortie déplacent des groupes de données à travers la frontière d'un utilisateur fonctionnel vers un processus fonctionnel et *vice versa*, en utilisant ou non un dispositif d'entrée/sortie (écran, clavier, souris imprimante ou autre). Les mouvements de Lecture et d'éCriture déplacent des groupes de données à partir du stockage permanent vers un processus fonctionnel et *vice versa*, peu importe le format de ce stockage permanent (table dans une base de données, fichier XML, mémoire ROM et autres).

Pour mesurer la taille, il faut compter un point par mouvement, par groupe de données, par processus fonctionnel compris dans la frontière. L'unité de mesure de la méthode COSMIC est le point de fonction COSMIC (PFC).

15.2.4 Démarche de mise en œuvre

Établir un référentiel de productivité du processus

Les organisations voulant mesurer la productivité de leur processus doivent d'abord établir un référentiel de la productivité d'un certain nombre de projets ayant appliqué sensiblement le même processus de développement.

Nous recommandons au moins une dizaine de projets pour commencer, afin de voir s'il y a corrélation entre la taille et l'effort, et entre la taille et la durée. S'il n'y a pas de corrélation significative, c'est en général parce qu'il existe un facteur différenciateur, comme la technologie, faisant en sorte que des processus différents ont été appliqués. Dans ce cas, nous suggérons d'augmenter la base de données des projets à quinze, vingt, voire trente projets, afin d'obtenir non pas un seul mais des modèles plus précis.

Pour établir ce référentiel, deux stratégies sont possibles, en fonction de l'importance d'obtenir rapidement cette valeur pour la direction :

1. Mesurer les nouveaux projets : les résultats ne seront connus qu'après avoir complété l'ensemble de ces projets ciblés ;

2. Mesurer un ensemble de projets récemment complétés : les résultats seront connus en quelques semaines seulement.

Dans un contexte de transition agile, il faut appliquer les deux stratégies, soit mesurer la productivité des projets faits avec l'ancien processus et mesurer la productivité de tout nouveau projet agile fait avec le nouveau processus.

Conseil : Assurez la qualité des données de taille fonctionnelle, d'effort et de durée afin qu'elles soient le reflet de la réalité de ce qui s'est passé, et non pas le reflet de considérations politiques ou budgétaires ayant peut-être mené à une distorsion des chiffres enregistrés.

Comment ?

Extraire les données d'effort et de durée des projets ciblés

Les organisations ont déjà en main les données d'effort et de durée de leurs projets. Toutefois, il ne faut considérer que la portion liée au développement logiciel, c'est-à-dire du démarrage du projet jusqu'à la mise en production des logiciels, incluant la gestion de projet, car pour comparer sur une base objective, il faut retrancher l'effort et la durée des activités de déploiement[1], comme la formation des utilisateurs et le développement du matériel de formation qui sont plus en relation avec le nombre d'utilisateurs impactés que la taille fonctionnelle. Alors, étant donné la comparaison souhaitée de la productivité des processus de développement, seuls l'effort et la durée du développement doivent être considérés.

Former des mesureurs ou faire mesurer par un expert

Les organisations doivent mesurer la taille fonctionnelle des logiciels développés ou modifiés par ces projets. Elles peuvent le faire à l'interne, à condition que ceux qui mesurent la taille aient reçu une formation adéquate. Si ceux affectés à la mesure de la taille sont tous débutants dans cette méthode, il est fortement recommandé de les accompagner pour vérifier les mesures de leurs premiers projets et ainsi augmenter leur savoir-faire. Pour établir le référentiel, nous recommandons de former une poignée de mesureurs (deux à huit personnes, selon le nombre de projets visés et la taille de l'organisation TI) ayant un profil d'analyste fonctionnel.

Les organisations hésitant à se lancer dans la collecte et l'analyse d'un nouvel indicateur de productivité peuvent demander à un expert d'établir leur référentiel en lui donnant accès à toute la documentation fonctionnelle des projets ciblés, qu'elle soit électronique, imprimée ou dans la tête des analystes. En fonction des résultats obtenus, ces organisations pourront plus facilement décider si elles veulent intégrer ce nouvel indicateur dans leurs données de gouvernance.

1. Se référer au chapitre *7 Déploiement* pour une liste des activités.

Constituer un référentiel de productivité des projets

Après avoir mesuré et vérifié la taille fonctionnelle des projets faits avec l'ancienne méthodologie, et ceux faits avec une méthode agile, il suffit d'entrer les données de taille, d'effort et de durée dans un tableur et d'en afficher les données graphiquement sous forme d'un nuage de points (Figure 15.6). Votre outil de tableur devrait pouvoir afficher les courbes de tendance sous forme de droite de régression linéaire ($y=ax+b$), accompagnées des équations correspondantes, comme illustré dans cet exemple. Ces équations représentent vos modèles de productivité, où :

- *a* est un nombre d'heures par PFC ;
- *x* est la taille fonctionnelle en PFC ;
- *b* est une constante qui doit être positive pour que le modèle soit valide, représentant *grosso modo* les coûts fixes des projets.

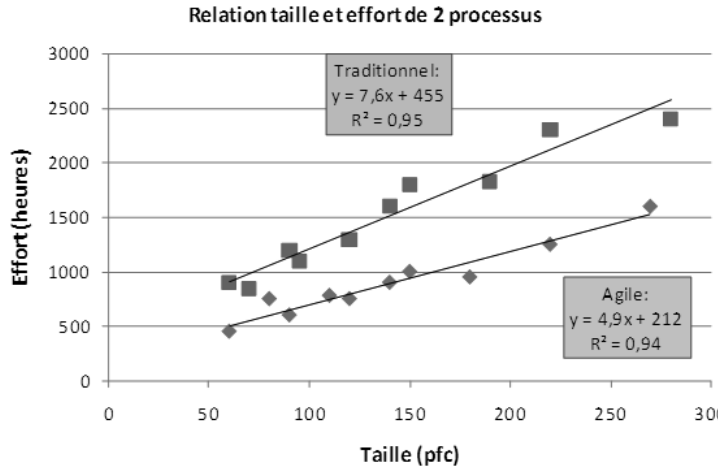


Figure 15.6 — Exemple de référentiel de projets fait en suivant deux processus, l'un traditionnel, l'autre agile.

Il nous est arrivé à plusieurs reprises de constater plusieurs modèles de productivité lorsque des technologies différentes mènent à des processus un peu ou complètement différents (par exemple, des projets sur des logiciels patrimoines en Cobol et d'autres, plus récents, en Java ou .NET). Ces cas se distinguent en général facilement sur un graphique en nuage de points. Il suffit alors de séparer les projets par technologie avant de réafficher le graphique. Il faut juste faire attention à ce que chaque sous-ensemble comporte un nombre suffisant de projets pour une analyse de qualité.

Comparer les résultats

Mise en situation : Jackie et la performance de son organisation (suite)

Jackie a fait mesurer le coût unitaire de plus de vingt projets. Puis, un étalonnage des résultats a été fait avec les données de l'industrie financière. Le coût unitaire moyen

s'est avéré de deux à quatre fois supérieur à celui de leur industrie. Ces données ont permis à Jackie d'amorcer un projet de réingénierie du processus de développement visant à en améliorer les coûts relatifs mais aussi les délais de livraison.

Un projet pilote d'adoption de l'agilité a été démarré avec une équipe ayant fait quatre projets avec la méthode Scrum. Les réductions relatives des coûts unitaires ont été les suivantes :

– Projet #1 : 13 %
– Projet #2 : 28 %
– Projet #3 : 40 %
– Projet #4 : 62 %

À partir du troisième projet, l'équipe atteignait déjà des performances supérieures à celles de leur industrie. Le projet pilote a été suffisamment concluant, au regard des objectifs ciblés, pour commencer à déployer l'agilité sur une plus large échelle.

Les premières comparaisons de la productivité peuvent être faites à l'interne, pour s'assurer que la démarche d'amélioration des processus augmente la productivité ou réduit le coût unitaire, si c'est l'objectif visé. Il est sain de vouloir comparer le coût unitaire de ses projets avec ceux de projets similaires dans le même domaine d'affaires et pour les mêmes technologies. Nous recommandons que ces activités d'étalonnage soient faites en second.

Combien ?

Pour construire un premier référentiel de productivité de dix à quinze projets, il en coûte de 10 à 30 jours d'effort d'un expert, plus environ 20 à 50 jours pour l'ensemble des mesureurs, répartis sur 3 à 8 mois.

La formation de base des mesureurs ne demande qu'une journée afin qu'ils soient capables de mesurer une majorité de fonctionnalités typiques de l'organisation. Cette formation de base doit être suivie de cinq à dix jours d'accompagnement par un expert qui vérifiera leurs données. Les mesureurs deviennent autonomes après qu'ils aient mesuré environ trois projets chacun et pour lesquels ils auront bénéficié de la rétro-information de l'expert.

L'effort de mesure prend moins d'une minute par PFC à un mesureur expérimenté lorsque les exigences sont claires et disponibles. Un mesureur débutant pourra passer jusqu'à cinq minutes par PFC. Lorsque les exigences ne sont ni claires, ni pleinement disponibles, l'effort de mesure peut grimper exceptionnellement jusqu'à quinze minutes par PFC puisque cela comprend l'effort requis pour chercher et trouver l'information nécessaire pour mesurer.

Sachant que la taille moyenne d'un projet varie entre 150 et 250 PFC, il en coûte de 2,5 à 20 heures par projet, selon que c'est l'expert ou le débutant qui mesure.

Lorsque les données sont cumulées et saisies dans un tableur, il en coûte moins d'une journée pour produire le modèle de productivité préliminaire. Il faut par la suite analyser les résultats et ajuster le modèle de productivité au besoin.

15.2.5 Estimation préliminaire des projets de développement

L'industrie constate que le talon d'Achille des méthodes agiles est leur difficulté à établir une estimation préliminaire de projet. Les équipes ont beau dire « *donnez-nous un budget et un délai, et nous vous dirons ce que nous pourrons accomplir* », il n'empêche que les dirigeants ont besoin d'avoir une idée des coûts globaux avant d'accepter de démarrer un projet. Lorsqu'un référentiel de la productivité du processus est établi, un gestionnaire ou une équipe peut se servir des données pour faire une estimation préliminaire d'un projet.

Une des techniques que nous utilisons est de se servir des données de mesure pour extraire la taille moyenne d'une nouvelle fonctionnalité et la taille moyenne d'une fonctionnalité modifiée. Ensuite, lorsqu'un nouveau projet est défini, nous n'avons besoin que du nombre de fonctionnalités nouvelles et modifiées qu'il a dans sa portée. Nous extrapolons la taille du nouveau projet avec ces données et lui ajoutons une contingence après avoir analysé les risques que l'analyse préliminaire soit incomplète. Puis nous appliquons l'équation retenue de productivité pour obtenir le nombre d'heures à budgéter.

Comparativement à une technique traditionnelle consistant à faire la liste détaillée de toutes les activités et à les estimer en heures, notre technique s'est avérée nettement plus précise, De plus, les chiffres budgétaires sont disponibles en quelques minutes, au lieu de plusieurs jours avec l'ancienne méthode.

15.2.6 Étalonnage : comparer sa productivité objectivement

Il est d'usage, dans certains domaines d'affaires, de faire l'étalonnage de la productivité de son processus de développement avec ceux de sociétés similaires. Un exercice d'étalonnage sérieux ne saurait se faire sans mesures objectives, répétables et reproductibles. Dans un tel contexte, il est nécessaire d'utiliser des mesures et des indicateurs comparables.

Au cours d'une transition agile, plusieurs dirigeants demandent à faire l'étalonnage de la productivité du processus traditionnel avec celle du nouveau processus agile, en général pour avoir un facteur de plus confirmant leur décision d'aller de l'avant avec l'agilité. Cependant, cet étalonnage interne ne dit rien quant à l'efficacité du processus dans leur domaine d'affaires. C'est pourquoi ces dirigeants voudront également faire l'étalonnage de la productivité de leur processus avec d'autres sociétés de même type.

La base de données ISBSG

Quelques firmes se spécialisent dans l'étalonnage pour des domaines d'affaires spécifiques. Leurs services sont coûteux, à la portée des grandes sociétés principalement. Comme alternative nettement moins coûteuse, il existe aussi une société sans but lucratif, l'*International Software Benchmarking Standards Group* (ISBSG[1]), qui fournit gratuitement un rapport d'étalonnage à toute société ayant soumis des données d'un

1. ISBSG : se prononce « ice bag », comme « sac de glace » en anglais, http://www.isbsg.org .

projet en utilisant l'un de leurs formulaires prévu à cet effet. Les données soumises sont systématiquement dénommalisées pour en assurer la confidentialité. Puis elles sont analysées pour leur cohérence et pertinence avant d'être intégrées dans la base de données de projets. Une nouvelle version de cette base de données est produite à intervalles de 12 à 24 mois.

À l'origine, ISBSG a été mise sur pied en 1998 par la province de Victoria en Australie qui cherchait un moyen et des données statistiques permettant l'attribution de contrats de développement logiciel en dollars par unité de taille, comme c'était le cas pour les contrats de construction routière. Devant l'absence de données fiables et transparentes, cette province en a financé le démarrage. Aujourd'hui, la base de données ISBSG contient des données de plus de 5 000 projets. ISBSG finance ses activités en vendant des licences de sa base de données ainsi que divers rapports d'étalonnage. Une personne détenant cette base de données peut en extraire les données de projets dont le contexte est similaire à ses projets – technologie, domaine d'affaires, type d'application – pour faire elle-même cet étalonnage.

15.2.7 Autres bénéfices à mesurer la productivité

En adoptant l'agilité, une société embrasse les principes d'inspection et d'adaptation dans un but d'amélioration continue. Elle veut pouvoir améliorer les aspects l'ayant poussé à adopter l'agilité mais aussi améliorer l'efficacité générale de ses façons de faire. Alors, les rétrospectives sont-elles efficaces ? Visent-elles à améliorer l'efficacité du processus et les communications ? La mesure de la productivité à travers le temps vient répondre au premier volet de façon directe et au deuxième volet de façon indirecte. Si les rétrospectives visent la bonne chose, les équipes verront la productivité de leur processus s'accroître.

Pour ceux qui aimeraient connaître le retour sur investissement de maintenir un groupe de soutien au développement, l'accroissement de la productivité sur une période donnée permet ce calcul, jusqu'à un certain point. L'augmentation de la productivité n'explique pas tout à elle seule et certainement d'autres facteurs doivent être considérés, comme le taux de roulement de personnel, la satisfaction du personnel et des utilisateurs, et la qualité des applications livrées.

15.3 MESURER LA QUALITÉ RELATIVE DES APPLICATIONS

Pendant longtemps, les informaticiens ont cru qu'une application est de qualité lorsque le nombre de défauts identifiés est presque nul ou lorsqu'ils ont démontré que l'application est conforme en tout point aux exigences spécifiées par écrit. Malheureusement, cela a parfois donné lieu à des applications répondant mal aux besoins car ce qui avait été mis par écrit n'exprimait pas bien les besoins réels. Avec l'adoption de l'agilité, le client faisant partie de l'équipe de développement fait en

sorte de diminuer la probabilité d'une mauvaise qualité à l'utilisation des applications développées, qui pourrait ne pas être liée au nombre de défauts identifiés.

La norme ISO 9126 – *Génie du logiciel - qualité des produits* – définit un modèle de la qualité et trois niveaux de caractéristiques et de mesures liées à la qualité :

1. **Qualité à l'utilisation** : efficience, productivité, sécurité et satisfaction, du point de vue des utilisateurs et du client.
2. **Mesures externes** : fonctionnalité, fiabilité, utilisabilité, efficacité, maintenabilité et portabilité, du point de vue des utilisateurs et des analystes.
3. **Mesures internes** : mêmes caractéristiques que les mesures externes mais avec un point de vue servant plus aux développeurs et aux architectes.

L'adoption de l'agilité visant l'augmentation de la qualité des applications devrait se refléter dans un plan de mesure où chacun de ces trois niveaux est considéré, et dans l'ordre. Ainsi, pour améliorer la qualité à l'utilisation, il est possible de vouloir mesurer la **satisfaction des utilisateurs**, typiquement *via* un sondage de satisfaction auprès d'eux. Toutefois, la gestion de la qualité des applications incombe aux responsables de ces dernières lorsque l'agilité est adoptée. Ces responsables d'applications sont les gardiens de la vision du produit logiciel et devraient déterminer les mesures et indicateurs leur permettant de constater l'atteinte des objectifs qu'ils auront fixés.

Dans un contexte où l'adoption de l'agilité s'est faite plus pour améliorer la qualité externe et interne des applications, notamment en tant que solution aux coûts croissants de correction des défauts, nous recommandons deux mesures dérivées impliquant la taille fonctionnelle :

- La **densité de défauts** : nombre de défauts par PFC ou par 100 PFC, à analyser dans le temps.
- Le **coût relatif des correctifs** : peut s'exprimer en heures de correction par PFC, à condition de cumuler l'effort de correction séparément. Cette mesure n'est mise en œuvre que lorsque tous s'accordent pour dire que la proportion d'effort passé en correction est significative et dérangeante. Si ce coût relatif des correctifs n'a pas été mesuré avant l'adoption de l'agilité, il est possible que cette mesure devienne non viable économiquement parlant après l'adoption de l'agilité si le problème est réglé, notamment par l'application des pratiques d'ingénierie agile.

15.4 AUTRES INDICATEURS UTILES

Les mesures dérivées et indicateurs décrits dans cette section sont des exemples que nous rencontrons lorsque les besoins en information les justifient. Par exemple, si l'adoption de l'agilité est née d'un besoin de réduire les délais de mise en production, il est normal de vouloir les mesurer et en suivre la progression dans le temps. S'il y a un nombre significatif de projets livrés au cours d'une période donnée, certaines organisations analysent la moyenne des délais de mise en production pour s'assurer

qu'elle baisse régulièrement, tandis que d'autres analysent les délais relatifs à la taille pour fins d'estimation.

Chaque besoin a sa solution, mais elle n'est pas toujours évidente ni rentable. Par exemple, si l'agilité est adoptée pour augmenter la qualité des communications et de la collaboration, il peut être ardu de mesurer cette amélioration si les éléments semblent intangibles. Néanmoins, voici quelques exemples de mesures et leur usage :

- Le nombre de scénarios utilisateurs acceptés ou refusés par séance de démonstration : l'analyse de tendance de cette donnée indique au client si son implication avec l'équipe est suffisante ou s'il doit s'investir davantage.
- Un sondage de satisfaction des membres d'équipe : alimentés par les spécialistes en gestion des ressources humaines, les résultats servent à ajuster les comportements de ceux interagissant avec l'équipe ou à ajuster certains aspects de communication dictés par les processus ;
- La quantité de points en suspens par projet : nous avons remarqué que les équipes ayant un haut niveau perçu de collaboration ont une faible quantité de points en suspens à gérer. Toutefois, une quantité élevée de points en suspens pourrait être circonstancielle à une complexité inhérente au projet et ne pas être le reflet de la qualité de la collaboration. Il faut donc être prudent dans le design et le choix des mesures afin qu'elles reflètent une situation réelle et dont l'interprétation ne fait aucun doute sur les conclusions à tirer. Dans le doute, s'abstenir de mesurer ou de présenter des résultats pouvant induire en erreur.

Lorsque vient le temps de mesurer des éléments plus concrets, comme de l'argent, le design d'une mesure vient plus naturellement. Par exemple, pour mesurer l'augmentation du retour sur investissement, les gestionnaires vont comparer les revenus générés par la nouvelle application et les coûts engendrés par son développement.

15.5 L'ASPECT ÉCONOMIQUE DES MESURES

Une mesure n'est pas gratuite, certaines mesures peuvent même s'avérer plus coûteuses que les bénéfices qu'elles apportent. C'est pourquoi la norme ISO 15939 recommande d'évaluer ses mesures régulièrement (étape 4 de la Figure 15.1).

Au moment de planifier une mesure ou un indicateur dans le plan de mesure, il est nécessaire de porter une attention à certaines des caractéristiques, comme les mécanismes de collecte, de vérification et d'analyse. En effet, les choix de mise en œuvre ont un impact sur le coût de la mesure. Par exemple, il est préférable d'automatiser la collecte des données pour en assurer la qualité, mais parfois les coûts de l'automatisation sont tels que la collecte et la vérification manuelles de quelques données demeure le choix le plus rentable.

Pour éviter d'inclure des mesures pouvant ne pas être utilisées, chacune d'elles doit avoir un consommateur et un usage bien identifiés. C'est auprès de ces consommateurs d'information qu'il faut vérifier si la mesure est toujours utile. Dans la négative, la mesure peut être retirée du plan de mesure.

Comme pour le développement agile, les items du plan de mesure devraient avoir une valeur d'affaires quantifiée en plus du coût annuel apporté par cette mesure. Ce calcul de retour sur l'investissement n'est pas si simple à déterminer puisqu'il doit inclure le coût des risques ou des conséquences découlant d'une mauvaise décision alimentée par des données de mauvaise qualité. De plus, pour éviter d'éliminer trop rapidement une donnée d'apparence inutile, il faut estimer le coût de ne plus avoir cette mesure.

15.6 LA FACE CACHÉE DES MESURES

Proverbe en mesure : « *Attention à ce que vous mesurez, car vous l'obtiendrez !* »[1]

Ce proverbe sous-entend que des effets pervers indésirables pourraient survenir. Par exemple, une organisation attribuant des bonus aux développeurs en fonction du nombre de lignes de code qu'ils produisent par année obtient du code plus volumineux à maintenir qu'une autre pour une quantité équivalente de fonctionnalités. Si une organisation décide de mesurer le nombre de défauts corrigés par développeur par année, sans aviser ce qu'elle fera de ces données, vous pouvez imaginer le type de comportement que ces développeurs pourraient avoir.

Les mesures font peur, particulièrement quand il y a un manque de transparence sur l'usage qui en découle ou, pire encore, lorsqu'elles sont utilisées à d'autres fins que celles pour lesquelles elles ont été conçues. Ainsi, mesurer la vélocité d'une équipe agile afin de se doter d'un mécanisme fiable d'estimation et de planification d'itération est très bien. Se servir des mêmes données pour pénaliser une équipe ayant moins bien performé qu'une autre apportera son lot de conséquences et d'effets secondaires non désirés, comme l'accumulation d'une dette technique[2], une attribution à la hausse du nombre de points d'effort par les équipes, et une démotivation des membres d'équipe se sentant jugés sur des critères subjectifs. De plus, des conséquences insidieuses peuvent naître de mauvaises pratiques de mesure dont la portée ne sera visible qu'après plusieurs mois, voire plusieurs années.

Pour éviter toute conséquence néfaste, il importe d'annoncer à l'avance l'usage et les décisions découlant des mesures, de rendre le processus de mesure transparent et démontrer cet usage. Bref, être cohérent entre ce qui est planifié, annoncé et exécuté. L'application des principes essentiels de la mesure devrait vous aider à déterminer les mesures appropriées et à appliquer un processus de mesure fiable.

1. Traduction libre de l'anglais « *Be careful what you measure, you will get it !* ».
2. Le sujet de la dette technique a été discuté au chapitre 5 *Équipes et livraison incrémentale*

En résumé

L'application des méthodes agiles repose sur l'inspection et l'adaptation fréquente de son processus logiciel. Quand les équipes n'utilisent pas de mesures en intrant à leurs rétrospectives, les constats sont qualitatifs et les améliorations parcimonieuses. Les équipes réussissant à améliorer leur productivité et leur qualité, tout en éliminant leur dette technique, utilisent immanquablement des mesures de performance dont les résultats nourrissent leurs besoins en information.

Les mesures de performance servent autant aux équipes qu'aux gestionnaires qui ont à cœur la performance de leur organisation. Les mesures de performance peuvent être peu coûteuses à maintenir lorsqu'elles sont analysées et améliorées par l'équipe de développement. Des méthodes normalisées existent, permettant une comparaison des résultats entre les projets et entre les organisations, notamment en mesurant la taille fonctionnelle des logiciels développés ou évolués par les projets avec la méthode COSMIC.

16

Agilité et CMMI

Objectif

Dans ce chapitre, vous découvrirez comment les pratiques agiles s'inscrivent dans une démarche d'amélioration des processus qui s'appuie sur un modèle d'évolution des capacités, en l'occurrence le CMMI. Vous constaterez que l'application disciplinée des pratiques agiles est conforme avec des pratiques du CMMI à tous les niveaux de maturité.

Mise en situation : Pierre et le CMMI – son contexte de départ

Pierre est le président fondateur d'une société qui offre des services de développement logiciel sur mesure dans le domaine financier. Ses clients lui fournissent suffisamment de travail pour occuper à temps plein son équipe de 13 personnes. Le carnet de commande comporte six mois de travail à venir. L'entreprise est profitable mais pourrait l'être plus : un projet sur deux dépasse son budget de 5 % à 37 %. Pierre pratique une offre de prix "plafond" sur chacun de ses devis, c'est-à-dire que le prix qui sera facturé n'excédera pas l'estimation établie, car il y voit un avantage compétitif. De plus, quand le client découvre des anomalies après le déploiement des fonctionnalités, Pierre s'engage à les corriger à ses frais, ce qui constitue également un autre avantage compétitif. Parfois, il soumissionne sur un projet de son client principal qu'il n'obtient pas, car un soumissionnaire d'un pays étranger [en voie de développement] offre un meilleur tarif horaire en plus « d'être CMMI[1] ».

(À suivre...)

1. À ce stade, Pierre ignorait ce que « être CMMI » signifiait lorsque son client le mentionnait.

16.1 LE CMMI

16.1.1 Qu'est-ce que le CMMI ?

L'acronyme CMMI signifie « *Capability Maturity Model Integration* » ou, en français, modèle intégré d'évolution des capacités. Le CMMI est une famille de modèles, appelés « *constellations* », comprenant les modèles suivants :

- CMMI *for Development* (CMMI-DEV), pour le développement de produits et services ;
- CMMI *for Services* (CMMI-SVC), pour la mise en place, la gestion et livraison de services ;
- CMMI *for Acquisition* (CMMI-ACQ), pour l'acquisition de produits et de services ;

Cette famille de modèles est définie par le Software Engineering Institute (SEI), l'organisme qui en orchestre le développement avec l'aide de très nombreux participants de l'industrie, le fait évoluer, et en assure la publication comme suit :

« Le CMMI est une approche de l'amélioration de processus qui fournit aux organisations les éléments essentiels d'une démarche efficace qui, en définitive, améliore leur performance. Le CMMI peut être utilisé pour guider l'amélioration des processus d'un projet, d'une division, ou d'une organisation toute entière. Il aide à intégrer des fonctions organisationnelles qui étaient traditionnellement séparées, à définir les objectifs et les priorités d'amélioration, à fournir une orientation pour la qualité des processus, ainsi qu'à fournir un point de référence pour l'évaluation des processus actuels. »

Dans ce livre, nous ne nous intéressons qu'au CMMI-DEV. Il s'agit d'un modèle des meilleures pratiques de développement de systèmes servant de guide à l'amélioration des processus. Lorsqu'il est appliqué au développement et à l'évolution de logiciel, il est référé comme étant un modèle de 'gestion du développement'. En effet, la majorité des pratiques qu'il contient font référence à de la gestion sous divers aspects. Ces pratiques de gestion constituent le noyau composant toutes les constellations des CMMI et elles représentent environ 80 % des pratiques de chaque constellation. Les 20 % résiduels sont composés des pratiques particulières et uniques à chaque constellation ; ainsi pour DEV il s'agit des pratiques catégorisées 'd'ingénierie'.

16.1.2 Ce que le CMMI n'est pas

Le CMMI n'est pas un processus. D'abord, une fois établi et documenté, un processus peut être directement appliqué par une personne, groupe ou organisation puisqu'il décrit « quoi et comment faire » alors que le CMMI ne propose que la description du « quoi faire ».

Par ailleurs, il ne faut pas voir le CMMI comme une liste de toutes les choses à faire absolument, incluant tous les détails explicatifs qu'il contient pour décrire ces choses. Les organisations ayant conçu un processus calqué sur chacune de ses pratiques ont créé un processus très lourd, une espèce de monstre difficilement applicable,

tout simplement parce que les membres des équipes se découragent dès qu'ils sont exposés à l'imposant volume de « documents » (parfois en papier, mais le plus souvent électroniques) décrivant comment ils doivent travailler. S'en suit alors une panoplie de demandes de dérogation étoffées pour échapper le plus possible au processus et pour arriver à livrer les fonctionnalités logicielles dans des coûts et délais raisonnables. Un tel exercice – d'avoir engendré un processus calqué point par point aux détails du CMMI – aura été un coup d'épée dans l'eau. Il est nettement préférable de définir le ou les processus requis, puis de vérifier si les pratiques du CMMI y sont mises en œuvre. Dans la négative, et seulement si l'atteinte des objectifs décrits dans le CMMI est compromise, des améliorations au processus seront considérées, estimées et priorisées.

Le CMMI n'est pas une norme. Les organisations démontrant une attitude dogmatique face à chacune des pratiques recommandées du modèle, agissent comme si chacune d'elles était obligatoire jusque dans ses moindres détails. Or, dans plusieurs contextes, de nombreux détails sous-jacents à certaines pratiques apportent peu de valeur. Un bon praticien doit se poser la question suivante pour chaque détail de pratique, dans chaque contexte : « *Et alors ? Que se passe-t-il si on ne met pas en œuvre ce détail de pratique ?* »[1]. Si la réponse est « *rien* » ou pire encore « *ça va diminuer l'efficacité du processus* », alors il est fort à parier que sa mise en œuvre doive être remise à plus tard. À la limite, une pratique pourrait être contre-productive si elle était mise en œuvre. Il est alors préférable d'utiliser une pratique alternative visant le même objectif. Certains pourraient considérer ces alternatives comme une mauvaise application des pratiques du CMMI mais, en réalité, elles sont une manière pragmatique de l'adapter aux besoins de l'organisation, ce qui est parfaitement conforme à la philosophie du modèle.

Le CMMI n'est pas une méthodologie. Il recommande des pratiques qui, lorsqu'elles sont appliquées, devraient permettre d'atteindre les buts cités dans le modèle. Il décrit « quoi » faire, pas « comment » le faire. Contrairement à une méthodologie, le CMMI ne décrit pas l'enchaînement des pratiques, ni quel acteur dans l'organisation doit les mettre en œuvre.

16.1.3 D'où vient le CMMI ?

Dans les années 1980, le *Department of Defense* américain (DoD, l'équivalent du Ministère de la Défense) était exaspéré par les dépassements de délais et de budgets des projets de développement de systèmes. Toutes les causes identifiées à ces dépassements pointaient les solutions logicielles. Était-ce donc si difficile de livrer du logiciel en respectant les coûts et les délais ? Comment le DoD pouvait-il réduire les risques d'un dépassement budgétaire ou d'échéancier des contrats qu'il octroyait ? Les réponses simples n'existaient pas. Le DoD a alors lancé un concours auprès des universités américaines pour mettre en place un institut qui relèverait le défi de faire du logiciel avec plus de rigueur. Ce fut l'Université Carnegie-Mellon (CMU) de Pittsburg qui

1. Anecdote humoristique : en dialecte québécois, cette question se traduit par « Wein pis ? », qu'on prononce « ouin-pi ». Lors d'une évaluation formelle des processus au regard du CMMI, nous nous amusions à surnommer certaines de ces pratiques par « les wein-pis d'Amérique ».

gagna et le *Software Engineering Institute* (SEI) y fut installé en 1984. Le DoD a ensuite demandé au SEI de développer un outil qui leur permettrait de qualifier le niveau de risque des sociétés auxquelles il envisageait l'octroi d'un contrat incluant du développement logiciel.

Le SEI a réuni une équipe de professionnels expérimentés en développement et en gestion du logiciel, sélectionnés d'après leur feuille de route de projets à succès et leur bonne réputation dans le domaine du logiciel. Ces experts ont identifié les bonnes pratiques des projets qui réussissent, c'est-à-dire livrant les fonctionnalités attendues à l'intérieur des paramètres de projets estimés : coûts, échéanciers et qualité.

Le SEI a lancé un questionnaire permettant de faire ressortir la maturité (ou robustesse) des processus des répondants aux appels d'offres. Ce questionnaire a servi pendant quelques années à améliorer la sélection des contractants pour le DoD, conduisant le SEI à publier une première méthode de diagnostic (*Software Process Assessment* ou SPA). Ensuite, le SEI a donné naissance à la première version du SW-CMM, le modèle d'évolution des capacités applicable au logiciel, ainsi qu'à deux méthodes d'évaluation de la maturité d'un processus logiciel. La première, le CBA IPI (*CMM-Based Appraisal for Internal Process Improvement*), est utilisé à l'interne par une société désireuse de constater l'état d'amélioration de son processus logiciel. La seconde, le SCE (*Software Capability Evaluation*), est imposée et réalisée par un maître d'œuvre à ses maîtres d'ouvrages dans le but de quantifier le niveau de risques à la réussite des projets logiciel. Le modèle SW-CMM comportait cinq niveaux de maturité. Dans chacun de ces niveaux, se retrouvaient des objectifs et des pratiques regroupés par « domaine de processus clés » (*Key Process Area* ou KPA).

Après quelques années de mise en œuvre, l'industrie s'est rendu compte que le SW-CMM était incomplet pour ce qui concerne le développement de systèmes matériels comportant des composantes logicielles. En parallèle, une série de modèles fut publiée par divers organismes dont un pour les « systèmes » mais qui n'était pas spécifique au logiciel et, surtout, qui n'était pas organisé par niveau de maturité. Il y avait également un modèle de travail d'équipe qui comportait des pratiques de composition et d'évolution d'une équipe afin de la rendre plus efficace. Il s'est avéré nécessaire d'harmoniser ces modèles pour toucher à l'essence du problème : livrer un produit complexe intégrant des composants matériels et logiciels. C'est ainsi qu'est apparu le CMMI, le « I » signifiant « intégration » de plusieurs modèles.

Dans sa représentation étagée, le CMMI, comme son principal précurseur le SW-CMM, comporte cinq niveaux de maturité, tel qu'illustré à la Figure 16.1. Le niveau « 1 » ne comporte aucune pratique, toute société qui fait du logiciel ou du développement de système peut qualifier d'emblée son processus au « niveau 1 ».

Le niveau 2, premier niveau regroupant des objectifs et des pratiques, met l'accent sur la gestion de projet. Dans l'évolution de son processus, l'organisation a d'abord structuré son processus de gestion du développement de produit pour chacun des projets, sans pour l'instant s'attarder sur les aspects d'ingénierie. Le niveau 2 accepte que ce processus ne soit pas nécessairement le même pour tous les projets ; théoriquement, à l'extrême, chaque projet pourrait fonctionner différemment, à la condition qu'au sein de chaque projet, un processus cohérent soit déployé. De plus, l'organisation

commence à définir, collecter et analyser des mesures et elle institutionnalise des pratiques de contrôle de conformité et de gestion de configuration.

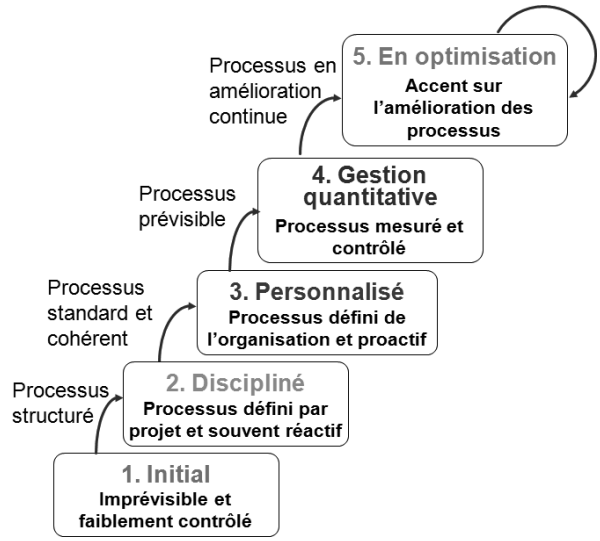


Figure 16.1 — Les niveaux de maturité du CMMI.

Au niveau 3, l'organisation a normalisé son processus intégré d'ingénierie pouvant être adapté aux besoins de chaque projet, à condition que la dérogation au processus standard soit justifiée. Ainsi, il y a une bibliothèque des actifs de processus alimentée et mise à jour par les projets. Le processus est évalué sur une base régulière. Un plan de formation organisationnel est constitué, exécuté, suivi et adapté. Les mesures et les indicateurs continuent d'être cumulés et analysés au regard du processus normalisé. Une gestion poussée des risques est appliquée, avec un plan de réduction et taxonomie, ainsi qu'une approche rationnelle et documentée pour aider à la prise des décisions majeures.

Pour passer du niveau 3 au niveau 4, l'organisation accumule et analyse une quantité significative de données statistiques sur la performance de son processus. À partir du niveau 4, les analyses statistiques tirées de ces données permettent une prédictibilité des performances du processus normalisé. L'accent est alors mis sur la qualité du processus, souvent en termes d'efficience et d'efficacité, ainsi que sur la qualité du produit, soit celle des logiciels développés ou évolués.

Quant au niveau 5, dit « en optimisation », l'accent est mis sur la gestion de changement. Le processus organisationnel normalisé est alors en amélioration continue.

16.1.4 Pourquoi utiliser le CMMI ?

Le CMMI doit d'abord et avant tout servir de guide à l'amélioration des processus de développement des systèmes. Il peut être utilisé pour évaluer ses façons de faire, situant ainsi le niveau de maturité de ses processus. À partir de là, il est possible d'établir un bilan pour ceux devant être adaptés et pour ceux méritant d'être améliorés en précisant pourquoi cette amélioration est nécessaire. Puis, il est possible de déterminer des objectifs ciblés, priorisés et bâtir la liste des actions à entreprendre pour les atteindre. Bref, une organisation doit gérer l'amélioration de ses processus comme n'importe quel autre projet. Même en mode agile.

Par la suite, à intervalles réguliers – de 6 à 24 mois, selon les objectifs d'amélioration souhaités et les budgets que l'organisation consent à allouer –, le CMMI est utilisé pour réévaluer la maturité des pratiques des processus, tel qu'appliqués dans les projets récents.

16.1.5 Les représentations du CMMI

Le CMMI est offert sous deux représentations : « en continu » et « étagé ». Les mêmes 22 domaines de processus sont présents dans les deux représentations, mais ils sont regroupés différemment.

Dans le premier cas, le regroupement est fait sous quatre catégories de « domaines de processus » (voir le Tableau 16.1). Dans le deuxième cas, les domaines de processus sont regroupés par niveau de maturité 1 à 5[1].

Pour que le processus d'une organisation soit évalué comme étant de niveau « *x* », elle doit démontrer qu'elle a mis en œuvre son processus, de manière soutenue et répétée, faisant en sorte qu'il y ait conformité avec tous les objectifs de tous des domaines de processus du niveau « *x* » et ceux inférieurs à « *x* »[2]. Les objectifs sont donc des éléments obligatoires. Les pratiques sont des éléments « attendus ». Bien que le modèle impose seulement des domaines et des objectifs, les pratiques sont fortement recommandées. À tout le moins, l'esprit derrière les pratiques doit être respecté.

16.2 MYTHES LIÉS À L'APPLICATION DU CMMI

Il est intéressant de constater qu'après plus de vingt ans d'existence des modèles d'évolution des capacités, certaines mises en œuvre douteuses ont donné naissance à toute une série de mythes, pouvant être qualifiées de légendes urbaines, puisqu'elles se sont parfois produites, au grand dam des personnes impliquées. Les mythes décrits ci-après sont ceux dont nous avons entendu parler le plus souvent. Si vous vous reconnaissez

1. Se référer au CMMI pour la représentation en étagé.
2. Par exemple, pour que le processus d'une organisation soit évalué au niveau 4, les pratiques du CMMI des niveaux 2, 3 et 4 devraient avoir été mises en œuvre de façon à démontrer que tous les objectifs de ces niveaux sont satisfaits.

Tableau 16.1 — Représentation « en continu » des domaines de processus par catégorie.

Gestion de projet	Ingénierie
Gestion des exigences Planification de projet Surveillance et contrôle de projet Gestion des accords avec les fournisseurs Gestion de projet intégrée Gestion des risques Gestion de projet quantitative	Développement des exigences Solution technique Intégration de produit Vérification Validation
Gestion de processus	**Soutien**
Focalisation sur le processus organisationnel Définition du processus organisationnel Formation organisationnelle Performance du processus organisationnel Gestion de la performance organisationnelle	Gestion de configuration Assurance qualité processus et produit Mesure et analyse Analyse et prise de décision Analyse causale et résolution

dans l'un ou l'autre de ces mythes, il est dommage que votre mise en œuvre du CMMI ait fait en sorte que vous passiez à côté d'opportunités extraordinaires. Dans un tel cas, peut-être que les explications données vous permettront de comprendre ce qui a pu se passer pour que vous en arriviez là et ainsi vous permettre de corriger la situation. Pour les autres, cette liste de mythes devrait vous être utile pour éviter certains pièges de mise en œuvre du CMMI.

Mythe n° 1 : Le CMMI ne s'utilise que dans les grandes organisations

Le CMMI est gratuit lorsque téléchargé du site du SEI, sinon il peut être acheté sous forme d'un livre, donc à la portée de toutes les bourses. Mais comment le SEI finance-t-il le développement et l'évolution du modèle ? De deux manières : via la formation certifiée donnée aux praticiens et aux évaluateurs, et via les *royalties* perçues à chacune des formations CMMI et, depuis peu, à chacune des évaluations « officielles », soit l'application de la méthode SCAMPI.

Toutefois, rien n'empêche une PME d'utiliser le CMMI comme guide d'amélioration des processus, sans pour cela chercher à faire évaluer le niveau de maturité de son processus. La PME un peu créative saura trouver comment mesurer les gains d'efficacité de son processus : meilleure qualité, meilleure productivité, meilleure satisfaction de la clientèle, meilleure maîtrise des estimations et des coûts de projets, meilleure rétention et satisfaction du personnel – les effets recherchés peuvent s'avérer nombreux.

Par où commencer alors ?

Mise en situation : Pierre et le CMMI – la mise en œuvre et les résultats

(Suite de la mise en situation en début de chapitre).

Un jour, lors d'un séminaire dans sa ville sur les meilleures pratiques d'ingénierie du logiciel, Pierre entend parler du CMMI. Il se demande alors si ce modèle d'évolution des capacités peut lui être utile. Il téléphone à la présentatrice dans les jours qui suivent pour discuter avec elle. Ensemble, ils abordent de la problématique de Pierre, de ce que peut apporter le CMMI à sa société, des coûts pour sa mise en œuvre, des avantages à espérer et des inconvénients à éviter ou amoindrir. Puis, Pierre décide d'entamer une démarche d'amélioration de ses processus en se servant du CMMI comme guide.

Pendant quatre ans, Pierre et son équipe optent pour une adoption itérative et incrémentale des pratiques du CMMI, tout en intégrant, chemin faisant, diverses pratiques agiles. Les résultats sont très positifs : sept projets sur huit respectent les budgets et les délais, moins d'un projet sur huit dépasse son budget d'une moyenne de 6 %, la satisfaction des clients augmente, le nombre d'anomalies post-déploiement est quasi nul, la productivité mesurée de l'équipe a triplé et son client principal lui octroie des contrats qu'il avait confiés, sans succès, en sous-traitance délocalisée.

(À suivre...)

Mythe n° 2 : L'application du CMMI complexifie le processus

Le CMMI ne dicte aucun processus. Le processus, c'est le vôtre, celui qui est appliqué par vos équipes. Cependant, lorsqu'une organisation investit dans la documentation de son processus, il est normal qu'il soit par la suite correctement appliqué. Sinon, il ne peut pas être qualifié de « processus mature ». Ce processus, sans égard à la complexité du domaine d'affaires et des risques associés à la non qualité des applications, devrait pouvoir s'illustrer sous la forme de flux de travail (*workflow*) et tenir sur une à cinq pages. Ce flux de travail devrait illustrer les phases, les étapes (séquentielles, parallèles, ou itératives), les points d'ancrages entre les phases, les rôles, les intrants et les extrants (aussi appelés « livrables » du processus). La documentation d'un processus doit également respecter les compétences des personnes qui l'appliquent : il ne doit pas expliquer leur métier en détail mais simplement illustrer l'enchaînement des étapes qu'ils doivent accomplir. Le processus doit demeurer léger : le documenter sous forme de pavé de 500 à 1500 pages que personne ne voudra lire serait considéré comme du gaspillage. À tous les niveaux, la règle des trois « U » doit s'appliquer : Utile, Utilisable, et Utilisé[1].

Mise en situation : Pierre et le CMMI – définir le processus

(Suite de la mise en situation précédente).

Au démarrage de l'utilisation du CMMI comme guide d'amélioration, le constat suivant a été fait : les projets de la société de Pierre ont tous le même cycle de vie. Le processus

1. Pour de plus amples informations sur la façon de documenter les processus en évitant les sept formes de gaspillage, vous référer aux chapitres *12 Évolution du processus* et *14 Agilité et documentation*.

est connu et appliqué de tous les membres du personnel, mais n'est cependant pas documenté, causant des pertes de temps quand vient le temps de l'expliquer à un nouvel arrivant. La société a donc décidé d'illustrer une vue d'ensemble contenant chacune des étapes et des points de contrôle de son processus, autant que possible sur une seule page.

Le résultat, disponible en seulement trois ou quatre demi-journées d'ateliers, a été très satisfaisant pour Pierre, car chacune des quatre phases de son processus est maintenant représentée sur une même page :

1. le démarrage, incluant l'estimation, la proposition et la planification,
2. la réalisation, incluant la gestion de projet,
3. la livraison, incluant les tests, le packaging et la préparation au déploiement ;
4. la fermeture, où les heures imputées au projet et la préparation de la facture au client sont vérifiées.

Sur chacune des pages, le processus, lisible, illustre chaque étape, les rôles impliqués, les intrants et les extrants, ainsi que les points de contrôle et les boucles de rétroaction. Le cycle de vie de chaque extrant permet d'identifier à quel moment les livrables sont introduits et comment ils sont bonifiés à chacune des phases. De plus, la forme attendue des livrables est documentée dans des gabarits, certains étant déjà disponibles, cumulés au cours d'années de pratique.

Au fil du temps et de l'avancement des projets, une quinzaine de gabarits ont été rattachés aux livrables du processus. La plupart sont petits, par souci d'efficacité, ne documentant que le minimum utile et nécessaire. Tous contiennent des instructions, au juste niveau de détail.

(À suivre...)

Mythe n° 3 : Chacune des pratiques du CMMI correspond à un livrable distinct du processus

Il est vrai que certaines organisations ont utilisé cette stratégie pour construire leur processus calqué sur le CMMI. Cela s'est avéré catastrophique du point de vue de la productivité et de l'efficacité. Ce qu'il faut comprendre, c'est qu'à chacune des pratiques du CMMI correspond une information qui devrait se retrouver dans un livrable. La forme importe peu : papier, document électronique, outil informatisé, etc.

Comment choisir les livrables à documenter ? Si vous développez déjà du logiciel qui fonctionne, alors vous avez nécessairement un processus, qu'il soit documenté ou pas, normalisé ou pas. Avant de revoir complètement vos documents, vérifier si vos livrables ne contiennent pas déjà l'information recommandée par le CMMI. Vous pourriez être étonnés du nombre de pratiques du CMMI qu'un seul livrable, plutôt simple, pourrait combler.

Mise en situation : Pierre et le CMMI – exemple d'un livrable utile

(Suite de la mise en situation précédente).

L'un de ces livrables, le plan organisationnel de vérification et de validation est sous forme de tableau, avec la liste des quinze livrables du processus dans la colonne de gauche, suivie d'une colonne pour chacun des items suivants :
– le rôle responsable de le créer ou de le modifier ;
– les rôles vérifiant sa qualité ;
– les critères de qualité à vérifier ;
– les techniques de vérification ou de validation à appliquer ;
– des guides ou listes de vérification ou de validation.
Ce livrable, à lui seul, tenant sur une seule page, comble 22 pratiques du CMMI, à condition qu'il soit appliqué systématiquement et intelligemment[1]. Il a suffi de réfléchir sur la façon d'être efficace au moment de sa conception. Les améliorations au processus de vérification et de validation ont consisté à bonifier les critères, les guides ou les listes de vérification et de validation.

Mythe n° 4 : Ça prend plus de dix ans pour mettre le CMMI en place

Le SEI a publié des données sur le temps moyen que les organisations mettent pour monter leur processus en maturité d'un niveau à l'autre. Pour les trois premiers niveaux, cela prend en moyenne de 18 à 36 mois par niveau. Les niveaux 4 et 5 sont moins ardus, car une majorité de pratiques ont été mises en place par les premiers niveaux. Alors, il est vrai qu'une organisation pourrait mettre jusqu'à dix ans pour atteindre le niveau 5. Toutefois, plusieurs organisations voulant avoir un processus performant de niveau 5, avec des mécanismes durables d'amélioration continue, y arrivent en quatre à six ans. Et si vous ne désiriez que contrôler votre gestion de projet pour commencer ? C'est-à-dire mettre en œuvre les pratiques de niveau 2, car elles viendraient combler les lacunes actuelles de votre processus. Dans ce cas, vous verriez des résultats en moins de six mois et l'atteinte de l'ensemble des objectifs du niveau 2 se ferait, selon toute vraisemblance, entre 18 et 36 mois.

En conclusion, mettre en place les pratiques du CMMI répondant aux problématiques les plus criantes ne prend que quelques mois. Il est toutefois recommandé de se faire accompagner afin d'assurer de le faire correctement.

Mythe n° 5 : Le CMMI et l'agilité ne s'appliquent qu'à des projets différents

Au début des années 2000 et à l'apparition des premières méthodes agiles, plusieurs organisations de développement logiciel ont prétendu appliquer ces méthodes agiles comme excuse pour l'application d'un processus complètement chaotique. Néanmoins, il est possible de démontrer qu'agilité et CMMI se marient très bien, qu'ils visent essentiellement les mêmes objectifs.

1. Il ne faut pas faire un livrable pour épater l'évaluateur CMMI mais bien parce qu'il est utile.

Dans la version 1.3 du CMMI, publiée en novembre 2010, des changements important ont été introduits au modèle pour expliquer la compatibilité des pratiques CMMI avec les principes agiles. C'est ce qui est décrit à la section 16.3.

Mythe n° 6 : Avec le CMMI, on documente tout, avec l'agilité, on ne documente rien

Nous avons entendu cette phrase à maintes reprises. Les gens peuvent avoir de mauvaises perceptions de l'un comme de l'autre. D'abord, le CMMI ne prescrit pas de tout documenter, ni quelle forme doit prendre la documentation lorsqu'elle s'avère nécessaire. Ensuite, les méthodes agiles décrivent largement ce qui doit être rendu visible, donc documenté sous une forme quelconque, tel que :

- Le carnet de produit et chacun des items qu'il contient et toute autre information jugée utile et nécessaire par l'équipe et le client, pour assurer la pérennité de l'application.
- Les différentes mesures et indicateurs : taille des items, effort restant, vélocité, résultats des tests, etc.

Comme vous pouvez le constater, les méthodes agiles produisent aussi des livrables documentés.

16.3 ARRIMAGE DE L'AGILITÉ ET DU CMMI

- Cette section du chapitre contient un résumé des résultats d'une analyse comparative détaillée faite par Sylvie à l'été 2006. La version CMMI-DEV 1.2 avait été utilisée. Ces résultats ont été présentés devant un public à deux reprises :
  - Conférence du SPIN[1] de Montréal le 26 septembre 2006 ;
  - Conférence de la FIQ[2] à Montréal le 25 octobre 2006, en collaboration avec François Beauregard, alors président de Pyxis Technologies et expert en agilité.

Deux ans plus tard, le SEI a publié une étude similaire intitulée « *CMMI™ or Agile : Why Not Embrace Both !* ». L'analyse qui suit a été mise à jour avec la version la plus récente du CMMI au moment d'écrire ces lignes, soit la version 1.3 du CMMI-DEV.

1. SPIN : *Software Process Improvement Network*, un regroupement de personnes ayant un intérêt professionnel dans l'amélioration des processus. La plupart des participants avaient une connaissance approfondie du CMMI et en étaient des praticiens.
2. FIQ : Fédération de l'informatique du Québec : une association de personnes ayant un intérêt pour l'informatique en général. Depuis, la FIQ a changé de nom pour le Réseau Action TI.

16.3.1 Pourquoi vouloir arrimer CMMI et agilité ?

Il y a plusieurs raisons à vouloir arrimer les pratiques du CMMI et celle des méthodes agiles :

- Lorsqu'une organisation ayant déjà adopté le CMMI souhaite bénéficier des avantages liés à l'agilité, sans compromettre la maturité acquise de ses processus.
- Lorsqu'une organisation a le souci d'adhérer à des pratiques sérieuses, rigoureuses, disciplinées et reconnues. Ses membres sont alors tentés par les pratiques agiles mais ils ont entendu toutes sortes d'histoires d'horreur liées à leur adoption, ce qui a eu pour effet de semer des préjugés défavorables envers l'agilité[1]. Dans ce cas, le fait de savoir que les pratiques agiles sont conformes aux pratiques du CMMI devient tout à fait rassurant.

16.3.2 Arrimage des pratiques agiles avec celles du CMMI

Au moment de faire l'arrimage des pratiques agiles avec celles du CMMI, quelques hypothèses ont été énoncées :

1. le contexte de développement logiciel est le suivant (fondé sur ce qui est rencontré le plus souvent) :
2. un projet = une version d'application = plusieurs itérations ;
3. une méthode agile de gestion de projet (ex. Scrum) est appliquée pleinement et sa mise en œuvre est soutenue depuis au moins six mois ;
4. une méthode agile de développement (ex. XP ou TDD) est appliquée et comprend les activités et livrables attendus pour la conception, le code, les tests, la documentation (utile et nécessaire) et la revue par les pairs.

Dès lors, une mise en correspondance de chacune des pratiques du CMMI a été faite avec les différentes pratiques agiles qui sont définies dans les méthodes choisies. Les résultats sont résumés ci-après et regroupés par catégorie de domaine de processus. Nous avons utilisé la légende suivante pour qualifier le niveau de conformité, pour chaque objectif du CMMI :

- PC : Pleinement conforme ;
- C : Conforme avec faiblesses mineures observées ;
- P : Partiellement conforme avec faiblesses importantes observées ;
- NC : Non conforme, hors de portée des méthodes agiles.

1. Toutes les histoires d'horreur que nous avons entendues sont liées à des organisations qui prétendaient appliquer des méthodes agiles alors qu'elles n'appliquaient que peu, voire aucune, des pratiques agiles de la méthode supposément adoptée.

Gestion de projets

Domaine de processus	Objectifs spécifiques du CMMI		Niveau de conformité
Gestion des exigences	SG1	Gérer les exigences	PC
Planification de projet	SG1	Établir les estimations	PC
	SG2	Développer un plan de projet	PC
	SG3	Obtenir l'engagement envers le plan	PC
Surveillance et contrôle de projet	SG1	Suivre le projet au regard du plan	PC
	SG2	Gérer l'action corrective jusqu'à terme	PC
Gestion des accords avec les fournisseurs	SG1	Établir les accords avec les fournisseurs	NC
	SG2	Se conformer aux accords avec les fournisseurs	P
Gestion de projet intégrée	SG1	Utiliser le processus personnalisé pour le projet	PC
	SG2	Coordonner et collaborer avec les parties prenantes pertinentes	PC
Gestion du risque	SG1	Se préparer pour la gestion du risque	NC
	SG2	Identifier et analyser les risques	P
	SG3	Réduire les risques	PC
Gestion de projet quantitative	SG1	Gérer le projet quantitativement	PC
	SG2	Gérer statistiquement la performance de sous-processus	C

Ingénierie

Domaine de processus	Objectifs spécifiques du CMMI		Niveau de conformité
Développement des exigences	SG1	Développer les exigences client	PC
	SG2	Développer les exigences produits	PC
	SG3	Analyser et valider les exigences	PC
Solution technique	SG1	Sélectionner les solutions de composants de produits	PC
	SG2	Développer la conception	C
	SG3	Implémenter la conception du produit	PC
Intégration du produit	SG1	Se préparer à l'intégration de produit	C
	SG2	Assurer la compatibilité des interfaces	PC
	SG3	Assembler les composants de produit et livrer le produit	PC
Vérification	SG1	Se préparer à la vérification	PC
	SG2	Réaliser les revues par les pairs	C
	SG3	Vérifier les produits de sortie sélectionnés	PC
Validation	SG1	Se préparer pour la validation	PC
	SG2	Valider le produit ou les composants de produit	PC

Gestion du processus

Domaine de processus	Objectifs spécifiques du CMMI		Niveau de conformité
Focalisation sur le processus organisationnel	SG1	Déterminer les occasions d'amélioration de processus	NC
	SG2	Planifier et implémenter les activités d'amélioration de processus	NC
	SG3	Déployer les actifs du processus organisationnel et y inclure les leçons apprises	C
Définition du processus organisationnel	SG1	Établir les actifs du processus organisationnel	P
Formation organisationnelle	SG1	Établir une capacité de formation organisationnelle	NC
	SG2	Dispenser la formation nécessaire	NC
Performance du processus organisationnel	SG1	Établir des référentiels de performance et des modèles	P
Gestion de la performance organisationnelle	SG1	Gérer la performance d'affaires	P
	SG2	Choisir les améliorations	PC
	SG3	Déployer les améliorations	C

Soutien

Domaine de processus	Objectifs spécifiques du CMMI		Niveau de conformité
Gestion de configuration	SG1	Établir des référentiels	C
	SG2	Suivre et contrôler des changements	C
	SG3	Établir l'intégrité	P
Assurance qualité du produit et du processus	SG1	Évaluer objectivement des processus et des produits de sortie	PC
	SG2	Fournir une image objective	P
Mesure et analyse	SG1	Aligner les activités de mesure et analyse	PC
	SG2	Fournir des résultats de mesure	PC
Analyse et prise de décision	SG1	Évaluer les solutions possibles	P
Analyse causale et résolution	SG1	Déterminer les causes des défauts	PC
	SG2	Traiter les causes des défauts	C

Objectifs génériques

Objectifs génériques du CMMI		Niveau de conformité
GG1	Se conformer aux objectifs spécifiques	Selon le domaine de processus
GG2	Institutionnaliser un processus géré	PC
GG3	Institutionnaliser un processus défini	PC
GG4	Institutionnaliser un processus géré quantitativement	PC
GG5	Institutionnaliser un processus en optimisation	PC

Pratiques du CMMI non couvertes par les méthodes agiles

À la lumière de cette analyse détaillée, vous comprendrez que la plupart des domaines de processus sont complètement ou partiellement couverts par les méthodes agiles, tandis que certains domaines de processus sont hors de leur portée. Typiquement, ces derniers sont ceux touchant le contexte organisationnel, bien au-delà des projets, tel que la formation organisationnelle, la gestion des accords avec les fournisseurs et la focalisation sur le processus organisationnel. Si une organisation adoptant l'agilité désire également faire évaluer ses processus au regard du CMMI, elle devra combler les pratiques manquantes en sus de celles apportées par les méthodes agiles retenues.

Autre constat important, aucune des pratiques agiles ne va à l'encontre des pratiques du CMMI.

16.4 GUIDE POUR ARRIMER CAPACITÉ DU PROCESSUS ORGANISATIONNEL ET AGILITÉ

En résumé, voici les points qu'une organisation doit avoir dans son processus, avec ses pratiques agiles, pour que ce processus s'élève en maturité :

- Une méthodologie de développement logiciel couvrant :
  - architecture, conception, gestion des interfaces, code, tests et revues par les pairs ;
- Un outil de gestion agile[1] pour générer et maintenir :
  - le carnet de produit ;
  - le carnet d'itération ;
  - les graphiques d'avancement du travail (*burndown*, vélocité, etc.) ;

1. Dans des contextes relativement simples, un tableur peut faire l'affaire. Dans des contextes le moindrement complexes, il est recommandé de se doter d'un outil à la hauteur de la performance requise. Le marché abonde d'outils de gestion agile.

- Des descriptions de processus organisationnels couvrant les pratiques manquantes des domaines suivants :
  - gestion des accords avec les fournisseurs ;
  - gestion des risques ;
  - gestion de configuration ;
  - focalisation sur le processus organisationnel ;
  - formation organisationnelle.
- Une méthode de rétrospective de projet ou d'itération couvrant spécifiquement les pratiques génériques suivantes :
  - collecter l'information d'amélioration (GP3.2, niveau 3) ;
  - assurer l'amélioration continue du processus (GP5.1, niveau 5) ;
  - corriger les causes des problèmes (GP5.2, niveau 5).

En résumé

Les pratiques agiles s'inscrivent tout à fait dans un processus logiciel mature et évolutif. Bien qu'un ensemble de méthodes telles que Scrum, XP, AM et TDD comblent un large éventail des pratiques préconisées par le CMMI, elles ne suffisent pas à toutes les couvrir. Une organisation se dotant d'une démarche d'amélioration et « d'agilisation » de son processus logiciel et désirant atteindre un niveau de maturité de 2 ou plus du CMMI, devra immanquablement combler les pratiques du CMMI non couvertes par les pratiques agiles, notamment celles ayant trait non pas à un ou des projets en particulier mais aux pratiques dites « organisationnelles », comme la focalisation sur les processus de l'organisation ou la formation organisationnelle.

17

Gestion des individus

Objectif

Dans ce chapitre, vous découvrirez les réactions que peut provoquer l'adoption de l'agilité sur les individus. Vous constaterez également comment les équipes de projets deviennent plus responsables envers la gestion du personnel, ayant ainsi des répercussions sur les pratiques des ressources humaines de l'organisation.

Mise en situation : L'évaluation annuelle de Marc-Antoine

Marc-Antoine, l'un des meilleurs développeurs de l'organisation a été convoqué par André, le directeur TI et Annie, la responsable des ressources humaines.

Il y a quelques mois, l'équipe de Marc-Antoine a décidé de pratiquer le binômage[1], pour encourager la conception en groupe, permettre de limiter l'introduction d'erreurs et diffuser les connaissances à travers les membres de l'équipe. Cependant, Marc-Antoine refuse catégoriquement de se plier à la pratique du binômage. L'équipe a fait appel à l'organisation car elle ne sait plus comment le raisonner. À la demande d'André, Marc-Antoine s'explique enfin :

– « *Les membres de mon équipe ont de bonnes intentions, mais je programme mieux qu'eux. Il n'y a pas de façon élégante de leur expliquer que je ne veux pas faire du binômage parce que je risque, moment de mon évaluation annuelle, de ne plus recevoir mon bonus habituel pour ma contribution exemplaire* ».

1. La définition de binômage décrite au chapitre *2 Les méthodes agiles* : pratique consistant à programmer à deux développeurs. Pendant que l'un code et explique sa démarche, le second fait de la revue de code en direct. Les développeurs intervertissent leurs rôles régulièrement.

Annie et André comprennent alors la position de Marc-Antoine. Ils le rassurent en lui assurant que l'organisation se penchera sur ce problème et s'engage à lui transmettre le plan d'action au courant du mois prochain. Marc-Antoine les remercie de leur compréhension et il accepte de pratiquer le binômage en attendant un retour de leur part.

17.1 RÉPERCUSSIONS SUR LES INDIVIDUS

Les précédents chapitres ont traité des impacts que peut avoir l'agilité sur les différents corps de métiers, notamment les architectes, les gestionnaires, les équipes d'infrastructure, les équipes de soutien, les responsables d'affaires et évidemment les équipes de projets. Bien que l'analyse de haut niveau permette d'identifier les impacts généraux du plan stratégique[1], chaque individu peut réagir d'une manière toute personnelle et différente à l'introduction de l'agilité.

Pour s'assurer de l'adhésion des membres de l'organisation, il est important de supporter la transition au niveau de l'individu. Cette analyse d'impact devrait être faite par un responsable proche de l'individu, par exemple son supérieur immédiat ou le leader de son équipe. L'objectif visé est d'accompagner l'individu en identifiant avec lui les changements envisagés, les attentes envers lui, les impacts possibles et ainsi l'aider à se tailler une nouvelle place pendant cette transition agile. Il est important pour l'accompagnateur d'avoir un sens de l'écoute afin de vérifier la compréhension de l'individu, de vérifier la compatibilité de ses intérêts personnels, de comprendre ses aversions et de mesurer sa capacité à intégrer le changement.

Alors que certains accepteront d'amblée l'arrivée de l'agilité, d'autres individus réagiront mal à son introduction. Certains seront détracteurs, d'autres auront simplement des difficultés à s'adapter et à l'extrême pourraient devenir des membres dysfonctionnels.

17.1.1 Les supporteurs de l'agilité

De notre expérience, la majorité des membres d'une organisation sont enthousiastes à l'idée d'adopter une méthode agile. Probablement parce que le *Manifeste agile* propose une alternative logique, qu'elle encourage une collaboration étroite et un engagement au succès des projets. Cet enthousiasme est contagieux et c'est souhaitable, parce que c'est la meilleure façon de propager la bonne application de la nouvelle approche et obtenir du succès.

Une organisation devrait communiquer les succès obtenus par les équipes et soutenir les initiatives et les bons comportements de ceux qui ont choisi d'adapter véritablement les approches agiles. D'abord, parce que c'est valorisant pour eux.

1. Le chapitre *11 Gouvernance* traite de l'introduction de l'agilité comme un des éléments du plan stratégique de l'organisation.

Ensuite, parce que c'est un outil pour transformer et convaincre les détracteurs et les membres dysfonctionnels de l'organisation.

17.1.2 Les détracteurs de l'agilité

Dans l'ouvrage *Succeeding with Agile*[1], l'auteur identifie quatre types de détracteurs pouvant apparaître suite à l'introduction de l'agilité : les *saboteurs*, les *irréductibles*, les *consentants* et les *sceptiques* (Figure 17.1). Les différents types de détracteurs se distinguent selon que les individus résistent activement ou passivement, et qu'ils ont une aversion pour les approches agiles ou préfèrent le *statu quo*.

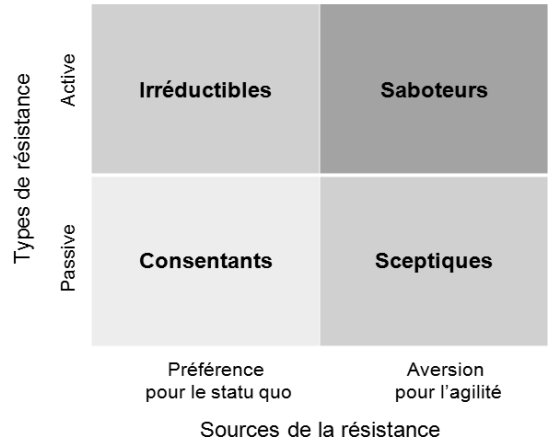


Figure 17.1 — Types de détracteurs d'une transition agile.

Les irréductibles

Ce type de détracteur tire habituellement avantage de la situation actuelle et il ne veut pas risquer qu'elle soit modifiée à son détriment. Il dénoncera l'arrivée de l'agilité et résistera à son introduction, prétextant que la situation actuelle convient et que le changement est inutile et sans valeur. Souvent leaders d'opinion, les irréductibles sont les rassembleurs de clans qui s'opposent au changement.

Voici des façons de gérer les irréductibles :

- Rendre la situation actuelle inconfortable, pour faire taire les arguments défendant le fait que la motivation de l'organisation à choisir l'agilité n'est pas justifiée. L'idée est de mettre en évidence un sentiment d'urgence pour convaincre les irréductibles[2].

1. *Succeeding with Agile : Software Development using Scrum* par Mike Cohn, Addison-Wesley Professional, 2009.

2. La motivation et le sentiment d'urgence sont des sujets de la gestion de la transition traités plus précisément au chapitre *10 Culture organisationnelle*.

- S'assurer de reconduire les intérêts personnels. Un irréductible est moins réticent lorsqu'il est rassuré que la prochaine situation est compatible avec ses intérêts personnels, par exemple : être membre d'une équipe qu'il aime, être l'expert d'un sujet précis, avoir de l'autorité et/ou du pouvoir, avoir des avantages et bénéfices accessibles pour lui.
- Désamorcer les appréhensions et les incompréhensions en expliquant les objectifs recherchés par l'agilité ainsi que communiquer le plan d'atténuation des impacts identifiés.

Les consentants

Les consentants n'ont pas d'intérêt particulier à changer. Ils veulent être certains que l'agilité n'est pas un effet de mode et que le changement sera durable avant d'emboîter le pas. Les meilleurs moyens de réduire la résistance des consentants consistent à montrer l'exemple :

- En les jumelant dans des équipes où le changement est accepté et où l'attitude est positive.
- En mettant l'accent sur les bons comportements, en particuliers ceux des détracteurs actifs qui acceptent graduellement le changement, plutôt que de concentrer l'énergie à modifier le comportement des consentants.
- En impliquant les consentants dans les activités de transition, comme la révision du processus organisationnel ou l'introduction de nouvelles pratiques comme le TDD ;
- En donnant l'exemple, en particulier de la part des coaches et des gestionnaires.

Comme pour les irréductibles, identifier les barrières naturelles comme la compréhension des raisons de la transition et les objectifs recherchés ainsi que vérifier les répercussions sur les rôles et la capacité des individus à s'adapter.

Les saboteurs

Les saboteurs refusent activement l'introduction de l'agilité, en enfreignant sciemment les règles ou en les interprétant avec de la mauvaise foi. Les façons de gérer un saboteur :

- Promouvoir les succès, afin de démontrer que l'agilité fonctionne bien dans le contexte de l'organisation.
- Encourager les communautés de pratique et les retours d'expérience pour que les saboteurs soient convaincus par des pairs plutôt que par la ligne hiérarchique.
- Diffuser l'intention et l'engagement inconditionnel de l'organisation face à la démarche de transition. Les saboteurs profitent habituellement des apparences de confusion pour attaquer le nouveau modèle.
- Les isoler en les affectant à une équipe qui n'est pas encore concernée par la transition à l'agilité.

Le licenciement reste la dernière alternative lorsqu'un saboteur n'entend pas raison, malgré tous les efforts. C'est un choix qui peut être difficile lorsque le saboteur est compétent et influent dans l'organisation. Il ne faut cependant pas sous-estimer le pouvoir qu'un saboteur peut avoir en ralliant une portion importante des consentants à sa résistance.

Retour d'expérience

Dans un contexte agile, une organisation avait décidé de se pencher sur le processus de rédaction des devis d'essai pour augmenter la valeur des documents et améliorer l'efficacité de leur production. Pour plusieurs raisons, ce processus fut long et laborieux.

Un des analystes, qui n'avait jamais eu foi dans la rédaction des devis, en a profité pour cesser la rédaction de ceux-ci, même dans leur forme actuelle. Après avoir tenté de le faire changer d'avis, son équipe avait fini par abdiquer à son opinion, jugeant que de toute façon, c'était lui le responsable de la rédaction des devis. L'organisation a fini par se rendre compte de l'absence des devis d'essai de l'équipe en question. Lorsqu'elle a demandé à l'analyste de s'expliquer, il a simplement répondu :

– « Le format des devis est en cours de révision. J'ai arrêté de les produire en attendant le résultat du format final. Il n'est pas question que je travaille sur un document dont le format est mal défini. »

Clairement, cet analyste était un saboteur qui avait profité de la moindre faiblesse du processus pour s'y abstraire, accumulant ainsi une grande quantité de travail non accompli. Après plusieurs tentatives pour le convaincre, l'analyste a finalement été écarté pour éviter qu'il répande son cynisme au sein de l'équipe.

Mathieu

Les sceptiques

Les sceptiques s'abstraient passivement des pratiques de l'agilité. Contrairement aux saboteurs, ils expliqueront leur retrait par le contexte de la situation plutôt que la confrontation. Par exemple, ils seront souvent en retard aux rencontres, expliqueront qu'ils ont oublié de vérifier la définition de terminé avant de clore le développement d'une fonctionnalité, ou ils oublieront d'écrire des tests unitaires, prétextant qu'ils ignoraient pour quels modules les tests étaient requis. Chaque événement inhabituel et tous les manques de précision du processus servent de prétexte aux septiques.

Le sceptique ne comprend habituellement pas les raisons et les objectifs du changement. Pour lui le changement représente une contrainte et il tente de s'en soustraire dans l'espoir que l'organisation reviendra à la raison en retournant aux anciennes pratiques. Le temps est le facteur le plus important pour convaincre les sceptiques. Avec le temps, il est possible de :

- former les sceptiques afin de leur communiquer les objectifs du nouveau processus organisationnel ;
- promouvoir les succès, afin de démontrer aux sceptiques le bon fonctionnement de l'agilité dans le contexte de l'organisation.

Un leader d'opinion sceptique peut être nommé comme LE sceptique officiel à qui l'on demande de jouer le rôle de l'avocat du diable. Ce sceptique a une opposition ouverte et connue. Son rôle est de soulever les problématiques introduites par l'adoption de l'agilité. Il est important que l'organisation recueille et traite sérieusement les problèmes identifiés par ce sceptique.

Les sceptiques peuvent également être convaincus lorsqu'ils sont eux-mêmes responsables de l'amélioration du processus. Par exemple, un exercice pour un sceptique ne croyant pas à l'intégration continue, pourrait être de le mandater à trouver des alternatives aussi efficaces pour assurer la non-régression de la qualité de la solution logicielle. L'adoption de la pratique sera plus efficace s'il a pu constater et apprendre par lui-même la valeur de cette pratique et l'absence d'alternative efficace.

17.1.3 Les membres dysfonctionnels des équipes

Il est possible que l'agilité fasse apparaître certains individus qui étaient devenus invisibles ou oubliés avec le temps. Cette section traite des types d'individus ayant du mal à s'adapter suite à l'introduction de l'agilité et que nous rencontrons parfois dans les organisations que nous accompagnons.

Retour d'expérience : Quand un membre d'équipe est dysfonctionnel

Nous avions entamé une transition agile dans une petite organisation développant des applications pour une grande société financière. Au cours d'une mêlée quotidienne de la première itération, un des membres de l'équipe, en poste depuis plusieurs mois, annonce aux autres qu'il a travaillé sur un truc fascinant. Les autres sont perplexes et l'un d'eux lui demande :

Membre d'équipe : Alors la composante logicielle que tu devais nous livrer est-elle prête ?

Dysfonctionnel : Forcément non, puisque j'ai travaillé sur cet autre truc vraiment fascinant.

Membre d'équipe : Oui mais nous attendons ta composante afin d'intégrer nos parties et vérifier que tout fonctionne. Maintenant, tu nous ralentis. Te serait-il possible de livrer tes engagements ?

Dysfonctionnel : Oui, bien sûr.

Malheureusement, le membre d'équipe dysfonctionnel a continué à travailler sur des trucs certes fascinants mais inutiles au reste de l'équipe. Après l'avoir formellement mis en garde à trois reprises et ayant perdu toute confiance envers ce membre d'équipe dysfonctionnel, l'équipe l'a éjecté. Ses talents et compétences auraient été utiles pour une autre équipe mais comme cette petite société n'en avait pas, le président l'a congédié.

Sylvie

Les individualistes

Certains individus sont solitaires et ont du mal à travailler en équipe agile. Souvent, ils sont seuls responsables d'un parc applicatif ou possèdent une spécialité particulière. Ils ont l'habitude de ne répondre qu'à eux-mêmes et ils sont maîtres de leur domaine d'intervention.

Même s'ils acceptent de rejoindre une équipe agile, ils ont tendance à s'isoler en s'affectant seulement des tâches ayant trait à leur domaine ou leur spécialité. Ce comportement est à proscrire car il va à l'encontre de l'esprit des équipes agiles. Le fait que tous les individus d'une équipe ne soient pas en mesure de prendre n'importe quelle tâche crée des dépendances et des goulets d'étranglement à l'intérieur de la gestion d'équipe. Une équipe dont la vélocité varie selon la présence de certains individus, où les membres ne sont pas en mesure d'estimer collectivement et où tous les membres ne sont pas en mesure de s'affecter n'importe quelle tâche a probablement des individualistes dans ses rangs.

La première chose à faire pour briser ce comportement est de partager le domaine de l'individualiste. Une fois que quelques membres seront capables de travailler sur le même domaine que l'individualiste, il sera possible de dégager du temps pour permettre à ce dernier de maîtriser de nouveaux sujets. À terme, la connaissance de tous les domaines de l'équipe sera diffusée entre tous les membres, formant ainsi une équipe multidisciplinaire.

Retour d'expérience : Quand le DBA passe de titre à pratique

Dans une équipe d'une dizaine de développeurs, le seul spécialiste des bases de données (DBA) se réservait toutes les composantes d'accès aux données. Il était très compétent et s'assurait de toujours créer des procédures stockées performantes pour traiter l'énorme volume quotidien de transactions. Il travaillait seul dans un bureau un peu à l'écart du reste de l'équipe.

Il a fini par devenir le goulet d'étranglement de l'équipe puisque les autres n'avaient pas ses compétences. Lors d'une rétrospective, l'équipe s'est dit que tous devaient apprendre du DBA pour être plus autonomes. Le DBA leur a donné quelques demi-journées de formation afin qu'ils soient capables de créer ou modifier les composantes d'accès aux données. Le DBA devait systématiquement faire une revue du code de ces composantes en plus de travailler en binôme pour les composantes plus complexes.

Après quelques semaines, la plupart des membres d'équipe étaient autonomes. Cela a permis d'augmenter la productivité de l'équipe en plus d'apprécier d'avantage les talents du DBA, maintenant moins solitaire et plus polyvalent.

Sylvie

Conseil : Ne remettez pas à plus tard la diffusion des connaissances pour des raisons de productivité. L'organisation désirant constituer des équipes multidisciplinaires doit protéger le transfert des connaissances, au détriment temporaire seulement de la productivité car elle augmente par la suite.

Les autoritaires

Les autoritaires ont typiquement un poste les rendant imputables des résultats d'une équipe. Leur motivation provient de la mise en œuvre d'un processus permettant d'assurer le succès des résultats. Ils utilisent leur autorité afin de protéger ce processus et empêchent les individus de prendre des initiatives mettant en péril les résultats d'un projet.

Dans le contexte d'une équipe autogérée, il y a de la place pour un leader responsable de motiver les membres et leur faire prendre conscience du sérieux des résultats. Cependant, il ne peut pas être directif au point de tuer l'initiative et être responsable de l'affectation des membres. Ce changement de mode de collaboration rend parfois la transition d'un chargé de projet vers celui de coach d'équipe difficile. Même conclusion pour un leader technique qui n'est plus le seul responsable d'un module ou d'une architecture.

Pour qu'un autoritaire accepte de changer, il faut d'abord lui retirer son imputabilité personnelle. Son équipe doit également faire preuve d'engagement pour éviter que son côté directif ne reprenne le dessus. Finalement, il faut trouver de nouvelles sources de motivation, autres que celle de la gestion des individus. Par exemple, la formation d'une équipe autogérée et multidisciplinaire avec l'objectif d'en faire une équipe ayant du succès dans l'organisation.

Les incompétents

La première caractéristique d'un incompétent est d'avoir du talent pour camoufler son incompétence. Il parvient à cacher son incapacité à livrer, soit parce qu'il est un beau parleur, soit parce qu'il parvient à blâmer d'autres individus, une situation, un système ou un processus. Ces raisons sont réelles et il est difficile de détecter les incompétents à moins d'appliquer le binômage. Le binômage permet d'apprécier les compétences des collègues et aussi de détecter les faiblesses, voire les incompétences chez les autres. Ça ne signifie pas pour autant que la personne doive être éjectée de l'équipe. Cela signifie plutôt qu'elle doit monter en compétences.

Habituellement, l'incompétence est connue, mais rarement mise à nue ou exprimée car les membres d'équipe adoptent souvent un comportement « sans blâme ». Les délateurs sont rares et personne ne veut être responsable de dénoncer l'incompétence d'un collègue. Cependant, avec l'introduction de l'amélioration continue et celle des équipes responsables, une équipe peut être amenée à identifier l'incompétence d'un membre comme la source d'amélioration la plus importante pour elle.

S'il y a peu d'incompétents, disons une faible minorité dans l'équipe, alors l'équipe mettra un certain temps à les détecter. La plupart du temps, si l'équipe arrive à compenser cette incompétence en prenant des engagements qui en tiennent compte, alors rien ne sera fait. L'équipe accepte d'endurer l'incompétence.

S'il y a beaucoup d'incompétents dans une équipe, disons la majorité, alors ça donne une équipe qui met beaucoup de temps à livrer ses engagements. Quand on fait un étalonnage de leur productivité, on constate souvent qu'ils sont significativement

moins productifs que d'autres équipes développant des applications similaires avec les mêmes technologies.

17.1.4 Gestion des membres dysfonctionnels par l'équipe

Comme traité à la précédente section, une équipe autogérée pourrait décider de l'expulsion d'un détracteur ou d'un membre dysfonctionnel de ses rangs. Nous reconnaissons cependant que c'est une action délicate qui demande d'être exécutée convenablement pour ne pas briser la bonne cohésion d'une organisation. Même si les praticiens agiles préconisent des ateliers et des techniques qui limitent le blâme et encouragent tous les participants à s'exprimer, les membres des équipes de projets ne sont pas des psychologues du travail.

Il est important d'accompagner ou former les coaches d'équipes pour les aider à bien réagir lorsqu'une situation humaine difficile se présente. Les coaches doivent également prendre conscience de leur limite d'expertise et savoir se tourner vers des experts lorsque nécessaire, comme les responsables des ressources humaines.

Conseil : Ne laissez pas traîner les cas d'individus difficiles à gérer car l'inconfort autour de ces personnes ne fera que grandir. Étudiez les solutions avec les responsables des ressources humaines afin d'appliquer celles qui conviennent à votre contexte.

17.2 IMPACTS SUR LES PRATIQUES DE LA GESTION DES RESSOURCES HUMAINES

Certaines pratiques normalement attribuées aux responsables de la gestion des ressources humaines (GRH) ou aux gestionnaires peuvent être impactées suite à l'adoption de l'agilité. Il est peu probable que ces impacts se manifestent au commencement de la transition. Ils apparaissent plutôt lorsque le processus agile est en place et qu'il est appliqué depuis quelques mois, voire des années. Il est tout de même possible d'identifier et planifier à l'avance certains impacts potentiels. La présente section décrit les impacts auxquels la GRH d'une organisation risque d'être confrontée.

Conseil : Les responsables de la GRH devraient être formés pour être en mesure de comprendre la dynamique et la philosophie de travail proposées par l'agilité. Cette introduction à l'agilité leur permettra d'identifier eux-mêmes les impacts possibles sur le processus de la GRH et de comprendre les demandes de l'organisation provoquées par la transition agile.

17.2.1 Description de poste

L'agilité peut introduire de nouveaux rôles dans l'organisation comme ceux du Scrum Master et du responsable de produit. Une organisation adoptant l'agilité pourrait vouloir créer de nouveaux postes au sein de l'organisation pour combler ces responsabilités. Parce que l'agilité introduit des rôles plutôt que des titres, nous ne recommandons pas de créer de nouveaux postes suite à l'adoption de l'agilité.

Prenons par exemple, le rôle de Scrum Master. Plusieurs individus différents peuvent avoir les aptitudes et les intérêts pour être Scrum Master :

- Savoir éliminer les barrières rencontrées par l'équipe ;
- Éduquer le responsable de produit à son rôle et lui démontrer comment il peut profiter des avantages et des bienfaits de l'agilité ;
- Utiliser le processus agile pour exposer les contraintes et les faiblesses de l'équipe ;
- Avoir le sens de l'écoute, questionner pour faire réfléchir l'équipe et montrer les conséquences ;
- favoriser la croissance de l'équipe, en la laissant prendre des risques calculés de manière autonome et lui donner l'occasion d'apprendre de ses propres expériences.

Les aptitudes d'un Scrum Master dépendent avant tout de son savoir-être, plutôt que de ses connaissances. D'ailleurs, les approches agiles sont des méthodes simples, épurées et facile à assimiler. Ce qui fait que n'importe quel professionnel avec le bon savoir-être pourrait devenir Scrum Master dans le cadre d'un projet, et retourner à sa spécialité dans le cadre d'un autre.

Même constat pour le responsable de produit, duquel il est attendu qu'il :

- Soit en mesure d'évaluer la valeur d'affaires du produit ;
- Soit capable de réunir les intérêts de toutes les parties prenantes du système ;
- Soit confiant et légitime afin de faire preuve de transparence avec le reste de l'organisation ;
- Ait les aptitudes pour diriger une équipe de gestion de produit ;
- Ait l'esprit d'entreprise afin de construire un système pour les utilisateurs et non simplement pour lui-même ;
- Sache se rendre disponible, soit ouvert et puisse communiquer et collaborer étroitement avec l'équipe de projet.

N'importe quel responsable d'affaires réunissant ces conditions peut jouer le rôle de responsable de produit le temps d'un projet, et retourner à ses fonctions habituelles par la suite. À moins que le métier de l'organisation soit de réaliser des produits logiciels, il est inutile d'avoir un membre permanent à la gestion de produit.

Transformation des rôles

Des postes actuels peuvent être modifiés suite à l'adaptation aux approches agiles du processus organisationnel. De notre expérience, les connaissances et les compétences des individus ne risquent pas vraiment de changer. Par exemple, un développeur restera un développeur malgré l'adoption d'une approche agile. Par contre, il se peut qu'il doive modifier son mode de collaboration pour être fonctionnel au sein d'une équipe agile. L'impact sur son poste se situera alors sur les attentes placées envers son savoir-être.

Voici la description des développeurs que nous avons rédigée dans le cadre de la transformation agile d'une organisation publique. Cette description est basée sur notre expérience personnelle, adaptée à la culture de l'organisation et complétée par l'ouvrage *Practices of an Agile Developer*[1].

La fiche descriptive du Tableau 17.1 décrit ce que l'organisation peut attendre de chacun des développeurs et ce, indépendamment des connaissances techniques et de l'expérience spécifique des individus. Une autre fiche descriptive a été incluse dans le processus de cette organisation pour décrire le rôle attendu de chacun des membres des équipes de projets.

La description du Tableau 17.2 décrit les attentes placées envers chacun des individus, et ce peu importe leur poste : analystes, développeurs, responsables de produit, responsable de l'assurance qualité ou encore Scrum Master. Les précédentes fiches de description des rôles démontrent bien que l'agilité affectera avant tout le savoir-être des individus. Par conséquent, c'est cet aspect des postes de l'organisation qui devra être adapté à la nouvelle réalité.

1. *Practices of an Agile Developer: Working in the Real World* par Andy Hunt et Venkat Subramaniam, Pragmatic Bookshelf, 2005.

Tableau 17.1 — Exemple de fiche descriptive du rôle de développeur.

Responsabilités	
Développer la solution logicielle ; Livrer la solution logicielle de manière incrémentale ; Estimer collectivement les stories du carnet de produit ; S'engager sur l'effort réaliste disponible pour chacune des itérations ; Démontrer les résultats obtenus au terme de chacune des itérations ;	Participer activement à la qualité du code produit (documentation, revue de code, propriété collective du code) ; Entretenir et adopter la définition de terminé ; Collaborer étroitement avec les analystes et le responsable de produit ; Tirer profit au maximum des outils et technologies utilisées.
Comportement attendu et compétences recherchées	
Utiliser et proposer la technologie en réponse à des besoins d'affaires, mais pas le contraire ; Produire des estimations et prendre des engagements (ex : estimation en points d'effort) : – qui sont justes et réalistes ; – qui demandent un rythme soutenu et soutenable ; – qui permettent une marge d'erreur acceptable avec l'intention de devenir une équipe prévisible ; Adopter des pratiques pour développer une solution simple et claire (ex : restructuration fréquente, statuer et implanter les solutions le plus tard possible, favoriser la réutilisation de code, diminuer le couplage et augmenter la cohésion des entités, etc.) ; Intégrer fréquemment son travail avec celui du reste de l'équipe projet (ex : en découpant le travail afin de limiter les grosses intégrations ou le verrouillage des fichiers) ; Assurer la qualité du code intégré avec celui du reste de l'équipe projet. Le code doit se compiler, s'exécuter, être testé et déployé ; Tirer profit des plans de tests automatisés pour : – diminuer les temps de rétroactions ; – augmenter la robustesse du code ; – augmenter le niveau de confiance envers le code et le produit ; – explorer l'utilisation de modules inconnus ; – découvrir les causes des défauts ; – documenter le code ;	Résoudre les problématiques et les défauts à la source, pour éviter qu'ils ne se reproduisent ; Documenter suffisamment : de manière à réduire au maximum le temps de compréhension avec l'effort d'entretien le plus faible possible ; Adopter des pratiques qui limitent le travail sans valeur ajoutée (ex : automatisation des tests, script de déploiement, ...) ; Adopter des pratiques qui favorisent la propriété collective du code (ex : revue de code, binômage, tests automatisés, conception en groupe, utilisation d'outils qui permettent le travail en isolation comme les sources control ou les wikis, etc.) ; Éduquer les autres développeurs lorsque nous maîtrisons une connaissance ou un concept ; Chercher à comprendre lorsqu'une connaissance ou un concept nous échappe ; Parfaire l'utilisation des outils et des langages de programmation en se documentant en ligne et dans les livres ; Faire des preuves de concept avec des conditions de succès préalables ; Se documenter sur les pratiques et les modèles utilisés et recommandés dans l'industrie (design patterns, utilisation des modules et des librairies, conventions de code et de langages, etc.).

Tableau 17.2 — Exemple de fiche descriptive des membres des équipes projet.

Responsabilités	
Rencontrer les objectifs du produit dans les paramètres de coût, de temps et de périmètre du projet ; Spécifier la solution logicielle ; Planifier la réalisation, basée sur les priorités d'affaires ;	Produire une solution logicielle qui rencontre les critères de qualité de l'organisation ; Communiquer l'état d'avancement du produit ; Communiquer les obstacles auxquels le projet fait face ; Documenter la solution logicielle.
Comportement attendu et compétences recherchées	
Être autonomes et auto-organisés ; Connaître la vision, les objectifs et les conditions de succès de son projet ; Faire preuve d'initiative ; Communiquer les obstacles et les avancées ; Prendre collectivement des engagements réalistes ; Collaborer étroitement avec l'équipe de projet et le reste de l'organisation ; Favoriser les canaux de communications efficaces (préconiser une courte discussion plutôt qu'un échange de documents, travailler à proximité dans la mesure du possible, ne pas attendre pour diffuser une information pertinente ou pour se renseigner à propos d'une information manquante) ; Être critique à propos des processus sans blâmer les autres ; Participer activement à la réalisation et aux rencontres ; Contribuer à l'efficacité (être ponctuel, se préparer avant une rencontre, donner son opinion, être ouvert d'esprit, se rallier aux décisions d'équipe) ;	Critiquer les rencontres qui n'atteignent pas leurs objectifs et proposer des alternatives ; Quitter les rencontres auxquelles nous n'apportons aucune contribution et dans lesquelles nous ne retirons aucun bénéfice ; Diminuer le temps de rétroaction et réduire le gaspillage ; Proposer des solutions moins coûteuses et plus simples lorsque possible ; Livrer la solution logicielle de manière incrémentale ; Valider les incréments fréquemment avec les utilisateurs du système dans un environnement représentatif le plus tôt possible ; Limiter la documentation aux besoins de ses consommateurs ; Préconiser les solutions simples et claires, plutôt que les solutions ingénieuses mais complexes ; Résoudre les problèmes à la source, pour éviter qu'ils se produisent à nouveau ; Ne pas croire en la pensée magique : les résultats ont peu de chance de changer lorsque les façons de faire sont les mêmes.

17.2.2 Plan de carrière

Avec la responsabilisation et l'autonomie des équipes, il est très probable qu'une organisation choisissant l'agilité aplatisse sa hiérarchie. Les organisations dites « plates » n'offrent pas de plan de carrière basé sur la promotion. Dans un modèle traditionnel, un développeur peut aspirer à devenir un leader technique, puis un chargé de projet, un architecte, ou encore un chef de département. Dans une organisation « plate », l'avancement hiérarchique est plus difficile, car il existe moins de postes de gestion.

Les processus GRH doivent prévoir de nouveaux plans de carrière pour ses collaborateurs. Des plans qui prennent en compte l'engagement de l'individu au sein de l'entreprise, la maîtrise grandissante de son expertise, sa capacité à élargir son champ de compétences afin d'être polyvalent, ainsi que son implication au soutien et à la réussite de son équipe. Cet avancement doit être orienté sur un rôle, et non pas sur un titre ou un poste hiérarchique.

17.2.3 Implication des pairs dans les processus de la GRH

Les approches agiles considèrent que le développement logiciel est avant tout un processus humain et que l'équipe joue un rôle clé dans le succès d'un projet. De plus, lorsque les individus sont maîtres de leur gestion, ils deviennent plus impliqués, plus engagés et ils sont plus efficaces pour résoudre les problèmes qui les concernent. À partir de ces principes, il apparaît normal que l'équipe soit impliquée dans les activités de GRH ayant un impact sur leur composition et leurs relations internes, comme l'affectation d'un nouveau membre, l'évaluation des individus, la rémunération et le retrait d'un membre.

Nous ne recommandons pas qu'une équipe soit seule responsable des activités de GRH. D'une part, le milieu de travail d'une organisation ne se limite pas simplement à celui d'une équipe. Les responsables GRH ont les compétences nécessaires pour évaluer le potentiel d'un individu ainsi que la compatibilité de ses ambitions avec les orientations de l'organisation. D'autre part, une culture du consensus appliquée à l'excès entraîne la tergiversation et empêche la prise de décision.

Malgré tout, une implication des coéquipiers dans les processus de la GRH est bénéfique et devrait être considérée dans un contexte agile.

Le processus d'embauche et la composition des équipes

> **Principe agile** : Bâtissez le projet autour de personnes motivées. Donnez-leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail

Au-delà d'un simple regroupement d'individus, une équipe efficace se doit d'être engagée et imputable collectivement de son succès. La formation d'une équipe efficace vise d'avoir des habiletés complémentaires, un but, une approche et un ensemble d'objectifs communs pour lesquels chacun se considère mutuellement imputable. Le choix des membres d'une équipe est un facteur important pour réunir des individus

ayant la chance de former un jour une équipe efficace. Non seulement ils doivent se compléter en connaissances et expertises, mais en plus, une chimie doit se développer entre les membres afin d'instaurer un climat de confiance.

C'est pourquoi les membres d'une équipe devraient être impliqués au moment d'embaucher un nouvel employé. L'équipe voudra s'assurer que les candidats partagent les mêmes valeurs qu'elle. Évidemment, la chimie n'est pas le seul facteur à considérer au moment d'embaucher quelqu'un. L'évaluation des compétences, la négociation salariale, la compatibilité entre le contexte offert par l'organisation et les ambitions du candidat sont autant de sujets pouvant être à la responsabilité de membres externes à l'équipe, comme les responsables de la GRH.

Conseil : Lorsqu'une organisation réussit à former une équipe performante, elle devrait dans la mesure du possible protéger la composition de cette équipe. La constitution d'une véritable équipe est un investissement demandant du temps et chaque mouvement à sa composition entraînera une régression de maturité. Lorsque le taux de roulement des membres d'une équipe est élevé, elle a peu de chance de devenir un jour une équipe efficace.

Le retrait d'un membre d'équipe

Comme traité à la section de la gestion des membres dysfonctionnels, une équipe peut être amenée à exiger le départ d'un de ces membres. Selon nous, une équipe ne devrait pas avoir à elle seule le droit de retirer l'un de ses membres. Premièrement, c'est un processus délicat qui peut briser la confiance de l'individu. Ensuite, ce n'est pas parce qu'un membre n'est pas fonctionnel au sein d'une équipe qu'il ne peut pas l'être dans une autre.

Les responsables de la GRH doivent être impliqués dans ce genre de décision. Cependant, une organisation qui a décidé d'adopter l'agilité sera confrontée un jour ou l'autre à gérer cette situation. Tout en encourageant les équipes à devenir autonomes et être plus efficaces, elle doit les soutenir dans la gestion des cas humains plus difficiles.

Évaluation individuelle

Le type de récompense influence grandement les comportements des individus. Si l'organisation évalue et récompense ses individus sur des critères de performance personnels, il sera très difficile d'y créer des équipes performantes. Par exemple, un individu évalué sur la qualité du code, la quantité de défauts identifiés ou encore le nombre de lignes de code écrites sera moins enclin à collaborer avec ses coéquipiers. Il sera plutôt motivé à livrer à lui seul une plus grande quantité d'objets sur lesquels sa soi-disant performance est mesurée.

Dans un contexte agile, le type de récompenses devrait se baser à la fois sur :

- l'expérience et l'expertise personnelle ;
- le savoir-être et l'implication de l'individu au sein de l'équipe ;
- la performance de l'équipe ;
- la contribution de l'individu au sein de l'organisation.

L'évaluation par les pairs est un moyen fiable d'évaluer la contribution et l'implication au sein des équipes et de l'organisation. Selon la maturité de l'organisation, cela peut se traduire par des évaluations 360°. Cependant, peu importe le niveau d'implication choisi, l'évaluation ne devrait pas se limiter à un seul entretien de l'individu avec son responsable hiérarchique et des responsables de la GRH. Dans un contexte agile préconisant la collaboration et l'amélioration continue, l'apport des coéquipiers a beaucoup de valeur ajoutée.

L'évaluation 360°[1] ne se limite pas à consulter les pairs de l'individu, mais aussi ses clients, ses fournisseurs, ses subordonnées et ses supérieurs. Dans une approche 360°, l'évaluation peut être faite et dirigée par un tiers, comme les responsables de la GRH, ou encore l'individu lui-même. Cet exercice a beaucoup de valeur pour l'organisation et l'individu, mais demande un certain degré de maturité et d'accompagnement pour être concluant et bénéfique.

Parce qu'elle définit des comportements, la description de rôle du Tableau 17.19 permet de définir les attentes que place une organisation envers les membres d'une équipe. Elle sert également de barèmes permettant aux pairs de les évaluer de manière objective et arbitraire.

En résumé

L'adoption de l'agilité est accueillie différemment par chacun des individus d'une organisation. Alors que certains acceptent le changement, d'autres réagiront de manière négative. Une organisation préparée est en mesure de mitiger les réticences aux changements qui pourraient mettre en péril sa transition à l'agilité.

Une fois adoptée, les attentes des individus envers l'organisation pourraient se transformer. Des individus engagés, au sein d'équipe autogérés, voudront à terme être impliqués dans certains processus GRH. C'est légitime, puisque les équipes sont imputables de leur performance et que les processus GRH ont un impact sur leur efficacité.

1. http://en.wikipedia.org/wiki/360-degree_feedback, consulté en mars 2011.

Conclusion

Adopter l'agilité vous transportera dans une merveilleuse aventure remplie de défis professionnels et humains. Malgré les défis et difficultés, la plupart des personnes sont enclines à embarquer dans l'aventure agile. Ceux ayant fait l'effort de la transition agile ne reviendraient pas en arrière, principalement à cause des gains récoltés à court terme.

Donnez-vous les chances de réussir. L'agilité est une philosophie de travail, alors faites-vous accompagner par des personnes ayant cette philosophie pour qu'ils vous influencent positivement vers votre réussite.

Préconisez l'amélioration continue de votre transition agile : appliquez des mécanismes d'introspection et d'adaptation chemin faisant. Identifiez les raisons de vos succès et trouvez des façons de rendre ces succès pérennes et reproductibles.

Mesurez votre succès en tenant compte des raisons pour lesquelles vous avez adopté l'agilité et des objectifs que vous aviez fixés : réactivité aux changements, délais de mise en marché, satisfaction des clients, satisfaction des collaborateurs, meilleure qualité ou productivité.

Nous vous souhaitons beaucoup de succès car nous pensons qu'aujourd'hui il n'existe pas de meilleure façon de faire du logiciel.

Mathieu et Sylvie

Références bibliographiques

Bibliographie

Agilité et pratiques agiles

• **En français :**

Claude AUBRY, *Scrum : Le guide pratique de la méthode agile la plus populaire*, Dunod, 2010.

Alistair COCKBURN, *Rédiger des cas d'utilisation efficaces*, Eyrolles, 2001. Note : Cet ouvrage est une traduction de *Writing Effective Use Cases* et est souvent cité comme étant LA référence en la matière.

Mike SULLIVAN, Steve JEWETT, et Curt HIBBS, *L'art du Lean Software Development*, Dunod, 2010.

• **En anglais :**

Lyssa ADKINS, *Coaching Agile Teams : A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition*, Addison-Wesley, 2010.

Scott AMBLER, *Effective Practices for eXtreme Programming and the Unified Process* –Wiley, 2002.

David J. ANDERSON, *Kanban : Successful Evolutionary Change For Your Technology Business*, David J. Anderson and Associates, avril 2010.

Kent BECK et Cynthia ANDRES, *Extreme Programming Explained : Embrace Change*, Addison-Wesley, 2004.

Alistair COCKBURN, *Agile Software Development*, Addison-Wesley, 2001.

Alistair COCKBURN, *Crystal Clear : A Human-Powered Methodology for Small Teams*, Addison-Wesley, 2004.

Alistair COCKBURN, *Writing Effective Use Cases*, Addison-Wesley Professional, 2000.

Mike COHN, *Agile estimating and planning*, Prentice Hall, 2005.

Mike COHN, *Succeeding with Agile : Software Development using Scrum*, Addison-Wesley Professional, 2009.

Mike COHN, *User Stories Applied : For Agile Software Development*, Addison-Wesley Professional, 2004.

Lisa CRISPIN et Janet GREGORY, *Scrum : Agile Testing : A Practical Guide for Testers and Agile Teams*, Addison-Wesley, 2009.

Rachel DAVIES et Liz SEDLEY, *Agile Coaching*, The Pragmatic Programmers/Pragmatic bookshelf, 2009.
Note : À la fin de chacun des chapitres de ce livre, il y a une liste de vérification des pratiques et comportements attendus d'un coach agile.

Esther DERBY, Diana LARSEN et Ken SCHWABER, *Agile Retrospectives : Making Good Teams Great*, Raleigh : Pradmatic Bookshelf, 2006.

Amr ELSSAMADISY, *Agile Adoption Patterns : a roadmap to organizational success*, Addison-Wesley Professional, 2008.

Michael FEATHERS, *Working Effectively with Legacy Code* – Prentice Hall, 2004.

Henrik KNIBERG et Mattias SKARIN, *Kanban and Scrum - making the most of both*, http://www.infoq.com/minibooks/kanban-scrum-minibook, consulté en décembre 2009.

Corey LADAS, *Scrumban : Essays on Kanban Systems for Lean Software Development*, Modus Cooperandi Lean Series, janvier 2009.

Mary et Tom POPPENDIECK, *Implementing Lean Software Development : From Concept to Cash*, Addison-Wesley, 2006.

Mary et Tom POPPENDIECK, *Leading Lean Software Development : Results Are not the Point*, Addison-Wesley, 2009.

Mary et Tom POPPENDIECK, *Lean Software Development : An Agile Toolkit*, Addison Wesley, 2003.

Andreas RÜPING, *Agile Documentation : A Pattern Guide to Producing Lightweight Documents for Software Projects*, Wiley, 2003.

Venkat SUBRAMANIAM et Andy HUNT, *Practices of an Agile Developer : Working in the Real World*, Pragmatic Bookshelf, 2005.

Ken SCHWABER, *Agile Project Management with Scrum*, Microsoft Press, 2004.

Ken SCHWABER et Mike BEEDLE, *Agile Software Development with SCRUM*, Prentice Hall, 2001.

Agilité à large échelle

• **En anglais :**

Pekka ABRAHAMSSON et Nilay OZA, *Lean Enterprise Software and Systems : First International Conference*, Proceedings of LESS 2010, Helsinki, Finland, October 17-20, Springer, 2010.

Jutta ECKSTEIN, *Agile Software Development in the Large : Diving Into the Deep*, Dorset House, 2004.

Craig LARMAN et Bas VODDE, *Practices for Scaling Lean & Agile Development : Large, Multisite, and Offshore Product Development with Large-Scale Scrum*, Addison-Wesley, 2010.

Craig LARMAN et Bas VODDE, *Scaling Lean & Agile Development : Thinking and Organizational Tools for Large-Scale Scrum*, Addison-Wesley, 2008.

Dean LEFFINGWELL, *Agile Software Requirements : Lean Requirements Practices for Teams, Programs, and the Enterprise*, Addison-Wesley, 2011.

Dean LEFFINGWELL, *Scaling Software Agility : Best Practices for Large Enterprises*, Addison-Wesley, 2007.

Thomas STOBER et Uwe HANSMANN, *Agile Software Development : Best Practices for Large Software Development Projects*, Springer, 2009.

Normes et modèles de bonnes pratiques

• **En français :**

Alain APRIL et Alain ABRAN, *Améliorer la maintenance du logiciel*, Loze-Dion, 2006.

Richard BASQUE, *Le CMMI : Un itinéraire fléché vers le Capability Maturity Model Integration*, Dunod, 2004.

Mary Beth CHRISSIS, Mike KONRAD, Sandy SHRUM, *CMMI 2e édition - Guide des bonnes pratiques pour l'amélioration des processus - CMMI ® pour le développement, version 1.2*, traduction en français approuvée par le SEI, Pearson Education France, 2008 - (ISBN 978-2-7440-7304-5).

Christian DUMONT, *ITIL pour un service informatique optimal*, 2e édition, Eyrolles, juillet 2007.

Organisation Internationale de Normalisation, Guide ISO CEI 2 : 2004, *Normalisation et activités connexes – Vocabulaire général*, 2004.

Organisation Internationale de Normalisation, Norme ISO 9000:2000, *Systèmes de management de la qualité – Principes essentiels et vocabulaire*, 2000.

Project Management Institute, *Le guide du corpus des connaissances en management de projet (Guide PMBOK)*, Project Management Institute, 4e édition, 2008.

• **En anglais :**

Hillel GLAZER, Jeff DALTON, David ANDERSON, Mike KONRAD, et Sandy SHRUM, CMMI® or Agile : Why Not Embrace Both !, Rapport technique, Software Engineering Institute (SEI), novembre 2008, pp. 48, http://www.sei.cmu.edu/reports/08tn003.pdf .

IT Governance Institute, *CoBIT 4.1*, 2007, disponible via http://www.itgi.org .

Organisation Internationale de Normalisation, *Software Engineering – Software Life Cycle Processes – Maintenance* (Ingénierie du logiciel — Processus du cycle de vie du logiciel — Maintenance), norme ISO/IEC 14764:2006, http://www.iso.org

Divers

• **En français :**

Oliver DEVILLARD, *Dynamiques d'équipes*, Éditions d'organisation, 2005.

René GOSCINNY et Albert UDERZO, *Astérix le Gaulois*, 6e édition, Hachette, 1999.

• **En anglais :**

Alain ABRAN, Software Metrics and Software Metrology, IEEE Computer Society, Wiley, Hoboken, NJ, États-Unis, 2010.

Marty CAGAN, *INSPIRED : how to create products customer love*, SVPG Press, 2010.

Reiner DUMKE et Alain ABRAN, *COSMIC Function Points : Theory and Advanced Practices*, CRC Press, Auerbach Publications, 2011.

Paul H. HERSEY, Kenneth H. BLANCHARD et Dewey E. JOHNSON, *Management of Organizational Behavior : Leading Human Resources*, 9e édition, Prentice-Hall, 2007.

Jon R. KATZENBACK et Douglas K. SMITH, *The Wisdom of Teams : Creating the high-performance organization*, Harvard Business School Press, 1992.

John P. KOTTER, *Leading change*, Harvard Business School Press, 1996.

Daniel PINK, *Drive : The Surprising Truth About What Motivates Us*, Penguin Group, 2009.

Ralph D. STACEY, *Strategic Management and Organisational Dynamics : The Challenge of Complexity*, Financial Times/Prentice Hall, 2002.

William E. SCHNEIDER, *The Reengineering Alternative : a plan for making your current culture work*, McGraw-Hill, Special reprint edition, 1994.

Standish Group, *Chaos report*, versions publiées entre les années 2004 et 2009.

Présentations

• **En français :**

François BEAUREGARD, *De l'agilité qui réduit l'agilité*, présenté à Agile Tour Montréal le 27 octobre 2010, disponible à http://www.slideshare.net/agilemontreal/agile-tour-montral-2010-de-lagilit-qui-rduit-lagilit-par-franois-beauregard-pyxis .

Jean-René ROUSSEAU, *Un coach agile : ça mange quoi en hiver ?*, présenté à Agile Montréal le 16 mars 2011, disponible à http://agilemontreal.ca/docs/UnCoachEnHiver.pdf .

• **En anglais :**

Emilio Di Zazzo et Douglas PAGE de CA Technologies, *Bridging the Gap Between Governance and Agility : Implementing Agile into your Organization*, présenté à Agile Montréal en novembre 2010, disponible à http://agilemontreal.ca/docs/Agile%20Montreal%20-%20Bridging%20the%20Gap%20between%20Governance%20and%20Agility%20-%20by%20Emilio%20Di%20Zazzo.pdf .

Martin PROULX, *Agile for managers*, Pyxis Technologies, 2010, disponible à http://www.analytical-mind.com .

VersionOne, *State of Agile Survey 2010*, obtenu à partir de http://www.versionone.com/state_of_agile_development_survey/10/default.asp, consulté en décembre 2010.

Webographie

• **En français**

Grand dictionnaire, http://www.granddictionnaire.com, consulté en janvier 2011.

L'internaute encyclopédie, http://www.linternaute.com/dictionnaire/fr/, consulté en juillet 2010.

Médiadico, http://www.mediadico.com/dictionnaire/definition/, consulté en juillet 2010.

Reverso, http://dictionnaire.reverso.net/francais-definition/, consulté en juillet 2010.

Sansagent, http://dictionnaire.sensagent.com/, consulté en août 2010.

Équipes autogérées : Gaillard conseil, *Stades de maturité d'équipes*, http://www.gaillard-conseil.com/, consulté en mars 2011.

Longueur des itérations : Tremeur Balbous, http://www.agilegardener.com/2010/04/22/les-points-de-visibilite-en-scrum-comment-choisir-la-longueur-des-iterations/, consulté en octobre 2010.

Maintenance de logiciel : S^{3M}, Modèle d'évaluation de la capacité de la maintenance du logiciel, http://www.s3m.ca/index_fr.html

Note : ce site web bilingue – anglais et français – regorge d'informations relatives aux bonnes pratiques de la petite maintenance de logiciel. L'abonnement de base est gratuit et donne accès à des publications, des outils et au site Wiki S3M.

Manifeste agile : http://agilemontreal.ca/fr/agilemanifesto_fr.php, consulté en novembre 2010.

MBTI : Wikipédia, « Le Myers Briggs Type Indicator (MBTI) est un test déterminant le type psychologique d'un sujet, suivant une méthode proposée en 1962 par Isabel Briggs Myers et Katherine Cook Briggs. », http://fr.wikipedia.org/wiki/Myers_Briggs_Type_Indicator, consulté en novembre 2010.

PATH : Pyxis Technologies, http://pyxis-tech.com/fr/notre-offre/coaching/path, consulté en février 2011.

Roue de la qualité : Wikipédia, http://fr.wikipedia.org/wiki/Roue_de_Deming, consulté en décembre 2010.

Sunset Graph : Elsa LARREUR, http://pyxis-tech.com/blog/2010/02/11/le-sunset-graph-comme-vecteur-de-communication/, consulté en janvier 2011.

• En anglais

Adaptation des contrats de sous-traitance de développement logiciel :

Martin FOWLER, http://martinfowler.com/bliki/ScopeLimbering.html, consulté en avril 2011.

Peter STEVENS, http://agilesoftwaredevelopment.com/blog/peterstev/10-agile-contracts, consulté en mars 2011.

Agilité et les équipes d'infrastructure : Scott WILSON, *Agile Operations*, http://agileoperations.net/, consulté en janvier 2011.

Agile UP : http://www.ambysoft.com/unifiedprocess/agileUP.html, consulté en octobre 2010.

Architecture agile : Andrew JOHNSTON, Agile Architect, http://www.agilearchitect.org. consulté en octobre 2010.

CMMI : Software Engineering Institue, *CMMI*, http://www.sei.cmu.edu/cmmi/, consulté en novembre 2010.

Coaching agile : Pierluigi PUGLIESE, *Solution Focused Approach to Agile Coaching*, Connexxo, http://www.slideshare.net/ppugliese/solution-focused-approach-to-agile-coaching-2604381, consulté en mars 2011.

COSMIC : Site internet de COSMIC, http://www.cosmicon.com, consulté en janvier 2011.
Note : Ce site comporte des articles, des manuels, des guides et des études téléchargeables gratuitement. Il y a également un guide de mesure COSMIC dans un contexte agile. Certains documents sont disponibles en français.

Culture et agilité : Michael SPAYD, *Agile & Culture : The Results*, Collective Edge Coaching : Leading Agile Transformations For The Enterprise, http://collectiveedgecoaching.com/2010/07/agile__culture/, consulté en décembre 2010.

Dette technique : Martin FOWLER, http://www.martinfowler.com/bliki/TechnicalDebt.html, consulté en octobre 2010.

Évaluation 360°, http://en.wikipedia.org/wiki/360-degree_feedback, consulté en mars 2011.

Équipes autogérées :

Martin PROULX, *Great news the project is over ! Now let's dismantle the team*, http://analytical-mind.com/2011/02/15/great-news-the-project-is-over-now-lets-dismantle-the-team/, consulté en février 2011.

Martin PROULX, *I don't believe in self-organized teams…*, http://analytical-mind.com/2010/08/30/i-don%E2%80%99t-believe-in-self-organized-teams%E2%80%A6/, consulté en mars 2011.

Gestion de la valeur acquise : Wikipédia, http://fr.wikipedia.org/wiki/Gestion_de_la_valeur_acquise, consulté en février 2011.

Graphique du stationnement : Mike GRIFFITHS, http://leadinganswers.typepad.com/leading_answers/2007/02/ summarizing_pro.html, consulté en janvier 2011.

ISBSG : *International Software Benchmarking Standards Group*, http://www.isbsg.org, consulté en novembre 2010.

Manifeste agile : http://agilemanifesto.org/, consulté en novembre 2010.

Modélisation agile : Scott AMBLER, http://www.agilemodeling.com/, consulté en octobre 2010.

OpenUP :

http://efp.eclipse.org/wiki/openup, consulté en novembre 2010.

http://en.wikipedia.org/wiki/OpenUP/Basic, consulté en novembre 2010.

Rôle du gestionnaire : Martin Proulx, président de Pyxis Technologies, *I dont't feel so good – I'm a people manager in an agile organization*, http://analytical-mind.com/2010/08/12/mommy-i-dont-feel-so-good-im-a-people-manager-in-an-agile-organization/, consulté en avril 2011.

Story Mapping: http://www.agileproductdesign.com/blog/the_new_backlog.html, consulté en octobre 2010.

Sunset graph : Martin PROULX, http://analytical-mind.com/2010/12/07/status-reporting-in-an-agile-context-%E2%80%93-welcome-to-the-sunset-graph/, consulté en janvier 2011.

Index

T

V

X