Le guide du petit Agiliste

Adrien CAUBEL

Préface

Adrien CAUBEL, je suis ingénieur en Science du Numérique dont trois années d'alternances à l'INP ENSEEIHT. L'ingénieur de demain sera amené à s'engager pour une ou plusieurs causes. Les miennes sont tournées vers l'éducation et la transmission du savoir.

Au travers le *Guide du petit Agiliste* je souhaite vous partager les concepts essentiels de l'agilité, expliquer l'importance de la culture Agile, et vous guider dans la réflexion de la transformation de vos équipes. Il sera par la suite nécessaire d'approfondir les concepts sur la philosophie Agile ou les cadres de travail associés au travers de d'autres ouvrages.

Ce livre n'est

- ni un cours détaillé sur l'agilité
- ni un cours détaillé sur Scrum ou Kanban

Agile n'est pas un boolean mais un float

Table des matières

S’assurer d’échouer

Comment échouer un projet ?

Ce premier chapitre vous donne des éléments permettant d'assurer l'échec d'un projet ou d'une équipe. La mise en lumière de ces aspects vise à sensibiliser les lecteurs aux pratiques et attitudes néfastes.

Le micro-management

Le micro-management se caractérise par un contrôle excessif, une surveillance constante et un manque de confiance envers les subordonnés. Ce contrôle rigoureux crée une culture de la peur et entrave une communication ouverte. Il empêche l'autonomie des équipes ce qui anéantit toute créativité et prise d'initiative. Tout le potentiel de l'équipe se voit inexploité.

Cette pratique contre-productive conduira à la fois à l'échec du projet et celle de l'équipe.

Agile devenu tabou

Ce n'est pas parce qu'une entreprise s'auto-étiquete Agile qu'elle facilite ou encourage les gens à travailler de cette manière. Toutes observations, remises en question ou critiques suggérant le contraire sont catégoriquement exclues.

Ces organisations se mentent et se porteraient mieux si elles assumaient leur fonctionnement et auraient pour réel objectif de l'améliorer.

Ne faite pas semblant d'être Agile

Aucune collaboration

Je ne prends aucun risque à dire que tous les projets logiciels nécessitent une collaboration des parties prenantes : experts métier, Product Owner, ingénieurs, graphistes, chefs de projet, testeurs, analystes, etc. Comme dans tout effort de collaboration, le résultat dépend de la façon dont toutes ces parties peuvent travailler ensemble.

- Sans feedback régulier des utilisateurs, le projet risque de diverger des enjeux initiaux.
- En l'absence d'un vocabulaire commun, nous courons le risque de rencontrer des incompréhensions concernant les besoins.

Augmenter la dette de motivation

Comparable à la dette technique, mais plutôt axée sur le manque d'engagement, de motivation ou de passion dans un projet. Si les membres de l'équipe ne sont pas suffisamment motivés, cela aura un impact négatif sur la qualité et l'efficacité du projet.

- Matériel inadéquat
- Gestion de projet bancale

Cette dette peut également être due à la qualité du code, on parle alors de mauvaise *Developer Experience*

- Technologie inadéquate pour répondre aux besoins métier
- Batailler pour essayer de comprendre ce que le code est censé faire
- Manque de documentation
- APIs non-intuitives

Promesse et illusions

Illusions

Réduire les coûts

L'approche Agile n'est pas destinée à réduire les coûts directs liés au projet, mais des économies peuvent être réalisées grâce à l'approche itérative et incrémentale. Les économies sont réalisées en livrant le bon produit, en récoltant les feedbacks rapidement et en apportant des ajustements.

L'investissement est optimisé pour concentrer les ressources sur ce qui apporte le plus de valeur aux clients et en évitant de gaspiller du temps et de l'argent sur des besoins qui ne sont pas essentiels et/ou qui ont évolué depuis le début du projet.

L'approche Agile ne promet pas de réduire les coûts

Réduire les délais de livraison

L'approche Agile ne garantit pas une livraison plus rapide. Cependant, elle essaie de conduire vers une livraison plus rapide, mais force est de constater que de nombreux facteurs impacteront les délais de livraison : courbe d'apprentissage, résistance aux changements, dépendances externes, etc.

L'approche Agile ne promet pas de réduire les délais

Boîte à outils

Vous n'être pas Agile parce que vous faîtes des cycles de deux semaines. Vous n'êtes pas Agile parce que vous faîtes un *daily*, une *rétrospective* ou une *review*. Devenir Agile ce n'est pas piocher des pratiques d'un framework (e.g. Scrum, Kanban). L'agilité est un état d'esprit qui guide le développement de logiciels.

Agile est une question d'état d'esprit et non de cérémonies

Facile à mettre en oeuvre

Effectuer une transformation Agile est une étape longue et difficile. Pensez qu'il suffit de copier-coller les cadres de travail Scrum ou Kanban vous amènera à l'échec et à une l'hostilité envers l'agilité.

L'agilité implique un changement drastique dans les organisations. Elles doivent traiter les problèmes de manière constructive, repenser le rôle des chefs de projet, agiliser la conception des produits et promouvoir une culture positive et constructive.

Agile ne fonctionne que si l'environnement et la culture les soutiennent

La Promesse

La promesse est unique,

Agile se concentre sur des fréquences de livraison rapide pour améliorer la satisfaction client. Ceci ce traduit par la construction d'un produit de manière itérative et incrémentale, en intégrant les clients dans l'équipe, en ayant une réaction rapide aux changements. Une pratique Agile tend à accroître la pertinence du travail effectué.

L'approche Agile vise à réaliser un produit qui correspond aux besoins du client

Une philosophie plus qu'une méthodologie

Mindset Agile

Avant d'entreprendre toute transformation au sein de l'équipe, il est essentiel de bien saisir les principes du *Mindset Agile*.

Le Mindset Agile doit faire partie de la culture d'entreprise

Changer votre vision

C'est reconnaître que le succès réside dans la capacité à s'adapter rapidement aux changements, à apprendre de l'expérience et à favoriser la collaboration entre individus et équipes. C'est également être prêt à remettre en question les normes établies, à rechercher constamment l'amélioration et à mettre l'accent sur la satisfaction du client.

En fin de compte, le *Mindset Agile* vise à favoriser l'innovation, à accélérer la livraison de produits ou de services de haute qualité et à créer un environnement où l'apprentissage continu est encouragé

Créer des acteurs

Les membres actifs se distinguent par une approche proactive et engagée envers le travail. Ces individus sont constamment à la recherche de moyens d'améliorer les processus, de résoudre les défis, et de collaborer de manière étroite avec leurs collègues pour atteindre les objectifs communs.

En adoptant un *Mindset Agile* vous encouragez la prise d'initiative, l'adaptabilité aux changements, et la volonté d'apprentissage constant. Vous créez un climat de confiance où chacun est libre de prendre la parole pour faire évoluer le groupe.

Il est donc également important d'accorder du temps pour l'exécution de tâches annexes, mais hautement importantes pour la survie du projet :

- Veille technologique
- Participation à des discussions inter-équipes au sein de l'entreprise
- Participation à des évènements et forums Tech

Transmettez le Mindset

Il faut de temps pour que les membres d'une équipe pensent Agile. Il faut prendre le temps de leur expliquer pourquoi nous faisons cette transition, il faut qu'ils prennent le temps de changer leur vision du Génie Logiciel. Pour faire ces changements, il n'y a rien de plus solide que de revenir à l'origine du mouvement en définissant ce qu'est une stratégie Agile puis en se référant au *Manifeste Agile*, qui contient quatre valeurs fondamentales et douze principes directeurs.

Vous n'arriverez pas à créer une équipe Agile, si les membres n'ont pas compris la culture Agile.

Stratégie Agile

Etre Agile c'est :

- adopter une approche itérative et incrémentale
- menée dans un esprit collaboratif
- pour générer un produit de haute qualité
- en tenant compte de l'évolution client

Itératif et incrémentale

Un processus *itératif* progresse par affinements successifs. L'équipe de développement met au point la première version du système, sachant à l'avance que certaines parties sont incomplètes. À chaque itération, les commentaires des clients sont pris en compte et le logiciel est amélioré par l'ajout de détails supplémentaires.

Un processus *incrémental* consiste à livrer des composants du logiciel par parties. Chaque incrément représente un sous-ensemble complet de fonctionnalités et est entièrement codé et testé.

La philosophie Agile combine à la fois la méthodologie incrémentale et la méthodologie itérative. Elle est itérative parce qu'elle prévoit que le travail d'une itération sera amélioré dans les itérations suivantes. Elle est incrémentale parce que le travail achevé lors d'une itération vient compléter la vision produit.

Esprit collaboratif

> " Travailler ensemble vers un objectif commun "

L'ensemble des parties prenantes (i.g membres de l'équipe, clients, parties intéressées) travaillent ensemble de manière transparente et ouverte pour atteindre les objectifs du projet.

Les membres de l'équipe partagent leurs idées, leurs défis et leurs réussites sans crainte de jugement, ce qui encourage l'innovation et l'amélioration continue. Les parties prenantes fournissent un feedback pré-

cieux tout en s'assurant que le produit répond aux attentes du client.

La collaboration ne se limite pas à la communication. On y inclut le partage des tâches, la résolution de problèmes et l'apprentissage mutuel. La diversité des compétences au sein de l'équipe est mise à profit pour maximiser l'efficacité et la créativité.

Haute qualité

" Notre priorité absolue est de satisfaire le client par la livraison rapide et continue de logiciels de qualité. "

La valeur est centrée sur le client et l'équipe de réalisation est responsable devant la satisfaction client. L'approche Agile marque ce tournant majeur dans la réflexion sur la construction d'un logiciel. Nous construisons un logiciel par itération, ce qui offre un feedback régulier afin de maximiser la valeur de la solution.

Évolution client

" Le client est au centre du processus de développement "

Lorsqu'on souhaite adopter une approche Agile il faut impliquer le client au quotidien dans le processus de création. En lui fournissant rapidement des *Incréments* nous nous assurons que le produit répond toujours aux attentes des utilisateurs et du client. Nous offrons ainsi au client une visibilité et nous nous protégeons en minimisant les coûts d'un changement de direction.

Le Manifeste Agile

Le Manifeste Agile est un document de référence dans le domaine du développement logiciel et de la gestion de projet. Il énonce un ensemble de principes (quatre) et de valeurs (douze) qui mettent l'accent sur l'adaptabilité, la collaboration et l'orientation client. Rédigé en 2001 par un groupe d'experts de l'industrie du logiciel, le manifeste sert de guide aux équipes et aux organisations qui cherchent à naviguer dans le paysage en constante évolution de la technologie et des affaires.

On y privilégie les individus et les interactions par rapport aux processus et aux outils, des logiciels opérationnels par rapport à une documentation exhaustive, la collaboration avec les clients par rapport à la négociation contractuelle, et l'adaptation au changement par rapport au suivi d'un plan.

Les quatre valeurs

Les premiers éléments		Les seconds éléments
Les individus et leurs intéractions	plus que	les procesus et les outils
Des logiciels opérationnels		qu'une documentation exhaustive
La collaboration avec les clients		la négociation contractuelle
L'adaptation au changement		le suivi d'un plan

Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers.

Les 12 principes

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.
3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
4. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
5. Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
6. La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.
7. Un logiciel opérationnel est la principale mesure d'avancement.
8. Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
9. Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.
10. La simplicité, l'art de minimiser la quantité de travail inutile est essentielle.
11. Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.
12. À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

Déclaration d'interdépendance

En 2005 de nombreux signataires du *Manifeste Agile* rédigent la *Déclaration d'interdépendance* qui se concentrer sur les principes de gestion qui permettent d'atteindre un *Mindset Agile* dans la gestion de projets.

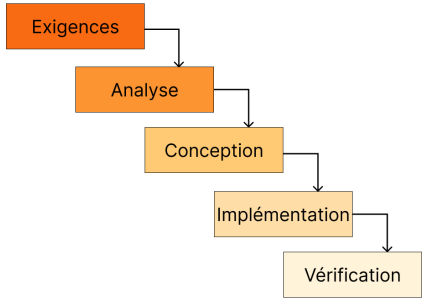
Résultat souhaité		Comment l'atteindre
Augmenter le retour sur investissement	en	Se concentrant sur les flux de valeur
Fournir des résultats fiable		Engageant les clients dans des interactions fréquentes
S'attendre à de l'incertitude		Gérant par itération, anticipations et adaptation
Libérer la créativité et l'innovation		Reconnaissant que les individus sont la source ultime de valeur. Créant un environnement dans lequel ils peuvent faire la différence
Stimuler les performances		Responsabilisant le groupe envers les résultats et l'efficacité de l'équipe
Améliorer l'efficacité et la fiabilité		Adoptant des stratégies, processus et pratiques spécifiques à chaque situation.

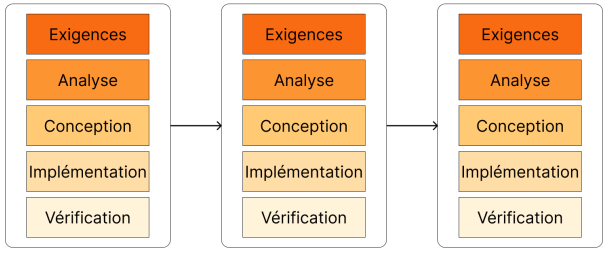
Gestion de projet Agile

Optimiser globalement

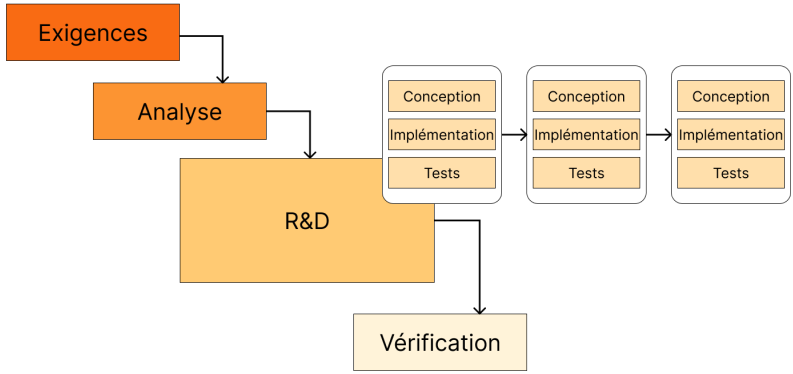
Lorsqu'on adopte une stratégie Agile il faut l'appliquer sur l'intégralité de la chaine de valeur. Lorsqu'une organisation fait l'erreur d'appliquer des cadres de travail localement elle ne peut pas en tirer l'ensemble des bénéfices.

Là où le *modèle en cascade* enchaine une séquence linéaire d'étapes avec très peu d'ajustement en cours de route l'approche Agile vise à produire plusieurs cycles composés de ces mêmes étapes.





Néanmoins, les grosses institutions peuvent avoir du mal à optimiser l'ensemble de la chaine. On se retrouve donc avec un modèle hybride où la chaine de production suit un processus cascade, mais où localement les équipes (e.g. R&D) adoptent une approche Agile.



De lors, il devient très difficile d'avoir un processus centré sur l'utilisateur, de recueillir du feedback client régulièrement et d'y apporter les ajustements à moindre coût. La rigidité organisationnelle va limiter l'efficacité de l'équipe de R&D.

Il est futile de dire que l'organisation adopte une approche Agile lorsque l'optimisation n'est que locale.

Équipe pluridisciplinaire

" L'équipe Agile doit disposer des connaissances nécessaires pour produire de la valeur au quotidien. "

Une équipe pluridisciplinaire se caractérise par un ensemble de personnes de disciplines différentes (e.g. marketing, développement, assurance qualité, etc) qui visent un objectif commun. Cette équipe permet d'être centré sur le client en encourageant une communication efficace au sein de l'équipe.

En évitant une organisation en silos, les équipes pluridisciplinaires se rendent indépendantes les unes des autres. On évite ainsi les goulots d'étranglement et l'attente de l'autre équipe qui pourrait remettre en cause les échéances du projet.

Il est important de noter qu'une équipe pluridisciplinaire n'est pas forcément une équipe dont chaque membre est expert de tout (*cross-skilled team*). L'idée est d'avoir une équipe qui puisse intervenir sur l'ensemble du périmètre grâce à ses membres qui ont une connaissance approfondie de leur spécialisation tout en ayant une connaissance superficielle des autres domaines. Si on regarde l'équipe dans son ensemble elle peut apporter une expertise profonde et durable.

L'équipe pluridisciplinaire minimise les sollicitations grâce à un fort spectre de connaissances

Équipe auto-organisée

Dans un environnement agile, le management délègue davantage de responsabilités aux membres de l'équipe.

Une équipe qui s'organise d'elle-même n'attend pas que les responsables leur confient du travail. Au contraire, elle identifie tout le travail à effectuer, hiérarchise les tâches nécessaires et gère elle-même les délais.

L'équipe dispose de l'autonomie nécessaire pour s'organiser de manière à accomplir au mieux les tâches

Ce que n'est pas l'auto-organisation

- L'auto-organisation ne signifie pas que les salariés conçoivent un modèle d'organisation.
- Elle ne signifie pas non plus qu'il faut laisser les gens faire ce qu'ils veulent.

L'auto-organisation signifie que la direction s'engage à accompagner les comportements qui émergent des interactions au lieu de spécifier à l'avance ce qu'est un comportement efficace.

Les équipes auto-organisées ne sont pas exemptes de tout contrôle de la part du management. Le management choisit pour elles le produit à construire et/ou choisit souvent les personnes qui travailleront sur leur projet, mais elles n'en sont pas moins auto-organisées.

Le besoin d'autonomie

Parmi les trois besoins psychologiques fondamentaux décrits par Deci et Ryan nous retrouvons le *besoin d'autonomie*.

Le besoin d'autonomie peut être défini comme la liberté de choisir, de planifier et de mettre en œuvre son comportement, ses actions, ses objectifs et son mode de vie. Donner de l'autonomie aux personnes est

un levier essentiel de motivation, et facilite l'engagement.

Au lieu d'avoir un responsable qui décide pour elles et de devoir adhérer à des politiques et protocoles stricts, les équipes disposent d'un certain degré d'autonomie :

- Comment les nouveaux membres d'une équipe sont recrutés.
- Comment les membres sont récompensés et évalués.
- Comment les membres sont formées.
- Comment les équipes synchronisent leur travail avec d'autres équipes.
- Les outils nécessaires pour effectuer leur travail.
- Comment les décisions sont prises au sein de l'équipe.
- Comment les équipes répartissent le travail.

L'auto-organisation est un symbole fort de la philosophie Agile. Viser l'auto-organisation c'est d'abord un management qui lâche prise et qui laisse une (grande) part d'autonomie aux équipes et aux individus.

Les managers donnent un cadre et soutiennent les équipes

Vocabulaire commun

Une des principales difficultés est que les développeurs ne sont généralement pas des experts métiers. Ainsi, lorsque le client énonce un besoin il s'adresse à un expert métier qui *traduit* le besoin pour les développeurs qui *traduisent* ensuite dans le code. Avec cette stratégie il y a de fortes chances que le message initial soit altéré et qu'on se retrouve à développer une bonne solution, mais sur le mauvais problème ou tout simplement à développer une mauvaise solution.

Une solution à ce problème est d'utiliser un *vocabulaire commun* (Ubiquitous Language) qui consiste à utiliser exclusivement les termes du vocabulaire métier. Ainsi, qu'ils soient ingénieurs logiciel, Product Owners, experts du domaine, concepteurs UI/UX ils utilisent un langage universel pour décrire le domaine d'activité.

Une communication efficace et le partage des connaissances sont essentiels à la réussite d'un projet logiciel

Transparence et visibilité

La transparence et la visibilité sont des facteurs importants pour la réussite du projet. L'équipe doit s'outiller pour fournir une visibilité sur l'avancement du projet et s'assurer que tout le monde est aligné sur les objectifs du projet.

Management visuel

Le management visuel permet de délivrer des informations et des indicateurs de manière simple et compréhensible. Par exemple, les tableaux Kanban, les burndown chart ou encore les user-stories, fournissent aux équipes des outils puissants pour communiquer, visualiser les progrès et identifier les problèmes.

Premièrement, la transparence fournie par cette approche permet aux équipes de réalisation d'avoir une compréhension directe du travail en cours. Ensuite, l'équipe est capable d'identifier les travaux importants et à forte valeur ajoutée. L'équipe adopte donc une approche proactive en identifiant rapidement le problème et en essayant de le résoudre.

Évaluer et adapter

La philosophie Agile encourage les boucles de feedback à différents niveaux. Ce retour d'informations aide les équipes à s'adapter et à faire les ajustements nécessaires.

Des réunions régulières de démonstration et d'examen sont organisées pour présenter le travail accompli aux parties prenantes (*review*). Ces réunions permettent d'obtenir un retour d'information et une validation, garantissant que le produit est conforme aux attentes. L'honnêteté et l'ouverture sont de rigueur, les informations sont partagées de manière proactive et tout obstacle ou problème est discuté ouvertement.

En interne, l'équipe se réunit également (*rétrospective*) pour réfléchir à leurs processus et identifier les domaines à améliorer. Cette pratique

encourage une culture d'amélioration continue et de transparence.

Puis au quotidien l'équipe se réunit (*daily*) afin de se synchroniser sur l'objectif de l'itération.

Motiver

La gestion des individus est au cœur de la culture Agile. La motivation des salariés est une condition nécessaire pour améliorer la performance économique (augmentation de la productivité) et sociale (diminution du taux d'absentéisme et du turn-over) de l'entreprise.

Collectivement

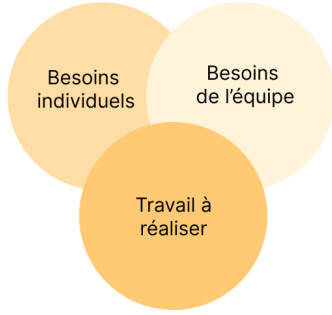
Dans gestion d'un projet Agile nous supprimons les rôles de contrôleurs et de décideurs. L'équipe devient responsable du produit et de sa bonne exécution. Elle peut ainsi innover dans le choix et la mise au point de techniques et pratiques.

Le management veille à créer un environnement de travail adéquat avec les objectifs fixés. Il respecte un rythme de travail soutenable et donne à l'équipe le pouvoir de s'autogérer.

Le management croit en la capacité de son équipe à atteindre les objectifs fixés et offre son soutien quand c'est nécessaire

Personnellement

Il ne sert à rien d'affecter a un salarié un projet qui ne représente aucune perspective de montée en compétences ou de motivation personnelle. Le management doit veiller à l'équilibre les besoins de chaque individu.



L'amélioration du processus

L'agilité accorde une grande importance au produit, elle reconnait également l'importance de l'amélioration continue du processus de création, en s'appuyant sur la philosophie Lean.
L'élimination du gaspillage doit être portée sur l'ensemble de la chaine de réalisation. On considère comme gaspillage

" Une perte est une consommation de temps/de cout qui n'apporte aucune valeur à un produit ou un service. "

L'objectif est clair, recherche la performance, la productivité et la qualité tout en respectant délais et cout. L'alliance des deux philosophies permet de former des acteurs capables de résoudre des problèmes et de faire face à des situations inattendues.

Brève introduction au Lean

Le *Lean Software Development* est l'application du Lean au développement logiciel pour une gestion au plus juste. On vise pour objectif d'obtenir pour l'activité du développement logiciel des résultats équivalents à ceux obtenus par les diverses applications du Lean (production industrielle, services, ingénierie, santé).

Le développement Lean est basé sur ce concept : élaborer une solution simple, la présenter aux clients et l'améliorer progressivement en fonction du feedback des clients. Pour ce faire il va falloir rendre les équipes autonomes pour soulever les problèmes et essayer de les résoudre.

Gestion des risques

Un risque est un événement ou une situation dont la concrétisation, incertaine, aurait un impact négatif sur au moins un objectif du projet.

La gestion des risques constitue donc un aspect crucial de tout projet. Dans un contexte Agile cette gestion se veut proactive et itérative en anticipant les risques tout au long du projet.

La gestion des risques est intégrée dans le pilotage quotidien du projet

Types de risques

Le Project Management Institute (PMI) classifie les risques en quatre catégories :

- Les risques techniques : logiciels, matériels, exigence de performance, ...
- Les risques externes : sous-traitants, réglementation, les clients, ...
- Les risques organisationnels : financier, politique d'entreprise, ressources humaines, ...
- Les risques de management du projet (interne à l'équipe projet) : communication, exclusion de certains membres, techniques de management, ...

Travailler par itération

En proposant aux équipes de travailler avec des cycles courts et des livraisons régulières nous pouvons facilement réévaluer les risques potentiels. Les équipes peuvent ainsi ajuster leur stratégie de gestion des risques à mesure que le projet progresse. Puis chaque incrément de travail offre une occasion de réévaluer les risques existants, d'en identifier de nouveaux et de mettre en œuvre des mesures correctives de manière continue.

Communication ouverte

En prônant la *transparence*, l'équipe se doit de partager constamment les informations sur les risques. Cela encourage la collaboration pour trouver des solutions et atténuer les risques.

On souhaite donc :

- Réduire le risque de perte d'engagement
- Réduire le risque de ne pas livrer les bonnes fonctionnalités
- Réduire le risque de livrer quelque chose qui a peu de valeur

Oser expérimenter

Être Agile s'est oser expérimenter, et encore plus, s'est laisser les équipes expérimenter

Apprendre de manière optimale implique une succession d'expérimentations. Plus on s'engage dans ces expériences, plus on accroît les probabilités de découvrir la meilleure idée. Pour être efficace, il est essentiel d'expérimenter rapidement, d'analyser promptement les résultats obtenus, d'assimiler les leçons tirées, puis de répéter ce processus.

Cette approche, définie comme l'amélioration continue, permet une progression constante en intégrant constamment les enseignements issus des expériences précédentes.

Échouer n'est pas un échec

Pour que les individus et les équipes osent expérimenter il est primordial qu'ils se sentent en sécurité. L'échec ne doit plus être perçu comme un échec, mais plutôt comme une étape essentielle vers le succès.

Le management doit encourager les expérimentations

Cadres de travail Agile

De haut en bas

Seulement et seulement une fois la culture Agile comprise et transmise vous pouvez envisager de déployer un cadre de travail (*framework*) dans votre équipe.

Toute mise en place d'un cadre de travail sans avoir pris conscience des modules supérieurs laissera place à de mauvaises interprétations et mènera l'équipe à une approximation des cadres de travail à minima. Cela sera contre-productif est pourra menée l'équipe à l'échec et les membres à rejeter de la culture Agile.

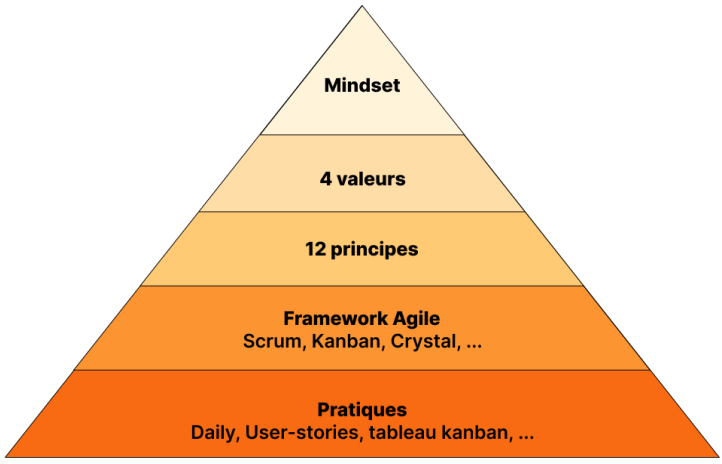


FIGURE 1 – Pyramide Agile

Le manager devient *un leader Agile*. Il devra accepter de perdre du contrôle sur l'équipe et faire confiance pour son autogestion. Il veillera à ne pas reprendre le contrôle lorsque l'équipe rencontrera des difficultés, mais à contrario la soutenir dans la résolution du problème. Il acceptera de ne plus attribuer les tâches en contournant le rôle de *Product Owner* et en transgressant la prérogative d'auto-organisation du développeur.

Une transformation Agile mal effectuée tournera au micro-management.

Scrum

Scrum, est un cadre de gestion de projet Agile. Il est devenu populaire parmi les organisations, en particulier celles qui cherchent à améliorer leurs processus de développement de produits. Il a été introduit au début des années 1990 par *Jeff Sutherland* et *Ken Schwaber*. Scrum est considéré comme un cadre de travail lourd qui s'articule autour d'un ensemble de rôles, d'événements et d'artefacts qui facilitent la création d'un produit correspondant aux attentes du client.

Scrum repose sur l'intelligence collective des personnes qui l'utilisent. Plutôt que de fournir aux gens des instructions détaillées, les règles de Scrum guident leurs relations et leurs interactions.

Il est profondément ancré dans les principes de l'empirisme. Ce mode de pensée met l'accent sur l'apprentissage par l'expérience et l'observation plutôt que de se fier uniquement à des plans ou des prédictions préconçus.

L'empirisme se traduit par trois piliers Scrum *Transparence*, *Inpection* et *Adaptation*. Ils garantissent que les équipes de projet évaluent constamment leur travail, procèdent aux ajustements nécessaires et s'efforcent de s'améliorer en permanence. L'équipe devient apte à répondre rapidement à l'évolution des besoins et de fournir de la valeur de manière plus efficace, ce qui est crucial dans le paysage commercial dynamique d'aujourd'hui.

L'apprentissage de Scrum commencera par la lecture primordiale du *Guide Scrum* [1] afin de comprendre les concepts et la terminologie de base.

1. https://scrumguides.org/download.html

Kanban

Kanban [2] est sans aucun doute l'un des cadres de travail les plus populaires après Scrum, rendu célèbre par son emblématique *tableau Kanban*, mais sa portée ne se limite en aucun cas à cette simple représentation.

David Anderson fondateur de Kanban avait pour intention de réguler les situations de sur charge de travail et les flux de demande variable afin d'équilibrer la capacité du système à la demande client et d'en réduire les variations. On retrouve également l'objectif d'équilibrer le travail non planifié du travail planifié et de trouver les bonnes priorités entre le travail piloté par les dates, les urgences et les tâches de fond.

Pour ce faire, Kanban abandonne les itérations au profit d'un travail en flux continu, en appliquant des améliorations progressives et en encourageant les équipes à évaluer le processus afin d'identifier et améliorer les domaines critiques.

Équilibrer la demande client avec la capacité via un travail en flux continu.

Tableau kanban

Le tableau Kandan permet de visualiser le système Kanban, il peut être physique ou virtuel l'essentiel est qu'il soit accessible à l'ensemble de l'équipe à tout moment. Il représente notre processus et est une photo à un instant t du parcours accomplit et à accomplir tout en fixant des limites. On notera que la plupart des équipes Scrum utilisent un tableau kanban sans aspirer au framework Kanban.

Ce n'est pas parce-que l'équipe à un tableau kanban, qu'elle fait du Kanban

2. https://kanbanguides.org/francais/

Scaler Scrum

Lorsqu'une seule équipe Scrum ne suffit pas à gérer la taille et la complexité d'un projet, nous devons adapter notre stratégie. Plusieurs cadres permettent d'appliquer efficacement les principes de Scrum dans des projets vastes et complexes en fournissant une approche de mise à l'échelle.

L'un des principaux problèmes de mise à l'échelle est les dépendances inter-équipes. Les frameworks de mise à l'échelle doivent donc veiller à réduire efficacement les dépendances inter-équipes

Nexus

Nexus est un cadre de travail introduit par Ken Schwaber basé sur les principes fondamentaux de Scrum. Nexus vise à étendre les principes agiles de Scrum à plusieurs équipes (3 à 9 équipes Scrum) travaillant sur un même produit, assurant ainsi une coordination efficace et une livraison intégrée à la fin de chaque Sprint.

Nexus est un groupe de trois à neuf Scrum Teams qui travaillent ensemble pour fournir un seul produit

Nexus ne modifie pas les rôles fondamentaux de Scrum (i.g. Scrum Master, Product Owner et Équipe de Développement). Cependant, il introduit le rôle supplémentaire de *équipe d'intégration Nexus* qui est chargée de faciliter la collaboration entre les équipes Scrum.

Comme l'ensemble des équipes Scrum travaillent sur un seul et même produit, Nexus propose d'utiliser **un seul Product Backlog** appelé *Nexus Product Backlog* qui est maintenu par **un seul Product Owner** et partagé avec l'ensemble des équipes. Cela garantit une vision commune des priorités et des objectifs de l'ensemble du projet.

Les rôles, les évènements, les artifacts d'une Scrum Team continuent d'être appliqués

Chaque équipe Scrum mène son propre Daily, effectue sa Rétrospective et organisation sa Revue de Sprint. Cependant, Nexus introduit également des événements supplémentaires spécifiques pour coordonner le travail entre les équipes :

- Nexus Daily Scrum qui est basé sur le concept du *Daily Scrum de Scrum* s'assure de la coordination et de la collaboration entre les équipes afin qu'elles progressent de manière synchronisée vers les objectifs communs du Sprint
- Nexus Sprint Planning qui coordonne la planification des Sprints entre les équipes, garantissant une compréhension commune des objectifs du Sprint et des éléments du backlog à traiter.
- Nexus Sprint Review et Nexus Sprint Retrospective similaires à la revue et à la rétrospective au niveau de l'équipe, sont conçus pour coordonner les revues et les rétrospectives à l'échelle du Nexus.

Ainsi, Nexus intègre des événements spécifiques pour coordonner le travail entre les équipes, tout en laissant intactes les pratiques individuelles des équipes Scrum. Une lecture du *Guide Scrum* est donc nécessaire avant d'entamer la lecture du *Guide Nexus* [3].

3. https://www.scrum.org/resources/online-nexus-guide

Pratiques Agile

Introduction

Une fois la philosophie Agile comprise et le framework choisi les équipes vont pouvoir alimenter leur système avec un ensemble de pratiques.

L'objectif des pratiques est de répondre de manière efficace et prouvée à un problème récurrent.

Les pratiques Agiles sont des Design Patterns de la gestion de projet.

L'ensemble des frameworks Agile utilisent/ont popularisé des pratiques. La plus connue est la réunion quotidienne (*daily*) qui répond au problème suivant : en raison des exigences émergentes et de l'incertitude, l'équipe de développement procède quotidiennement à une nouvelle planification afin d'augmenter les chances d'atteindre les prévisions.

A noter que dans ce chapitre nous ne reviendrons pas sur l'ensemble des pratiques existantes (plusieurs centaine [4]). Les plus populaires sont décrites dans les frameworks (e.g. Scrum Guide).

4. https://sites.google.com/a/scrumplop.org/published-patterns

Affinage du Product Backlog

L'affinage du Product Backlog n'est pas une cérémonie imposée dans les cadres de travail Scrum ou Kanban. Néanmoins elle s'est largement popularisée et devient une réunion presque systématique et périodique dans les équipes Agile.

Durant l'affinage du Product Backlog les personnes nécessaires se réunissent autour du Product Owner dans le but de décomposer les éléments du Product Backlog en éléments plus fins et plus précis. À la fin de cette réunion, on souhaite avoir une estimation affinée de la valeur d'un élément et une réorganisation de notre Product Backlog. D'autres informations pourront s'ajouter suivant les conversations menées comme les critères d'acceptation.

L'affinage du Product Backlog est une activité qui demande une forte intensité mentale. Ainsi on se consacrera en priorité sur les éléments qu'on estime implémenter dans les prochaines itérations.

Les User Stories

L'user-story s'est imposée comme une norme pour rédiger des exigences fonctionnelles. La rédaction d'user-stories efficaces est une compétence cruciale. Elle vise à ce que l'équipe comprend les besoins de l'utilisateur et qu'elle puisse fournir le plus de valeur.

Les user-stories seront sujet à discussion, il est important que l'ensemble de l'équipe est une compréhension commune

Granularité d'une user-story

Chaque acteur voit l'user-story avec une granularité différente. Pour le client, elle représente une fonctionnalité qui permet d'atteindre ses objectifs, pour l'utilisateur elle accomplit un besoin, pour un développeur elle prend quelques jours pour être implémentée.

Ensuite, on attend souvent le terme d'*Epic*. Chaque équipe apporte sa propre définition, mais nous retiendrons ici celle de Mike Cohn

" Une épopée (epic) est une grande user-story. Il n'y a pas de seuil magique à partir duquel nous qualifions une story particulière d'épique. Cela signifie simplement *big user-story*. "

Une épopée correspond souvent à une fonctionnalité principale du produit, ou une nouvelle idée, dont les détails ne seront précisés ultérieurement lorsqu'ils seront nécessaires. Elle se décomposera en une ou plusieurs user-story(ies)

Qui rédige une user-story ?

Majoritairement, les user-stories sont définies et rédigées par un expert du métier (e.g. Product Owner, Business Analyst). Il s'assure que l'user-story est en adéquation avec la vision du produit ou de release.
Néanmoins, chaque membre de l'équipe d'implémentation (e.g. développeurs) peut rédiger des user-stories. Ils devront les soumettre au Product Owner qui se portera garant ce nouveau récit.

Quand rédiger les user-stories ?

Il existe plusieurs moments ou une user-story peut être rédigée.

1. En amont le Product Owner a pu rédiger une liste d'*éléments* nécessaires pour le client. Il a pu les exprimer sous la forme d'une user-story, mais sans que ce soit une obligation.
2. Durant les réunions d'affinage du backlog, l'équipe se retrouve et affine les éléments pour qu'ils deviennent des user-stories.
3. Il faut juste être conscient que l'user-story doit être complétée et finie avant la phase d'implémentation.

Comment rédiger une user-story

Les principes INVEST et 3C offrent des concepts essentiels pour rédiger des user-stories efficaces.

INVEST représente le *but*, on souhaite que nos éléments répondant à la typologie *Independent*, *Negotiable*, *Valuable*, *Estimatable*, *Small*, et *Testable*.

Un des *moyens* d'y arriver est d'utiliser les ateliers *User Story Mapping* en passant par les 3 phases *Carte*, *Conversation* et *Confirmation*.

- Avec la Carte on souhaite retranscrire les points importants de notre pensée. Puis lorsqu'on regarde la carte, on se remémore l'ensemble de la Conversation. La carte est un point d'ancrage.
- Durant la Conversation nous allons clarifier l'histoire : à qui s'adresse cette fonctionnalité et que veut-il dedans, discuter des problèmes sous-jacents, poser des questions, apporter un point de vue critique. À la fin de la Conversation, l'équipe a une vision claire et commune.
- Finalement, elle conserve ses accords (Confirmation) en écrivant les critères d'acceptation sur la carte.

User Story Mapping

Si j'exprime une idée que j'ai en tête et vous demande simplement si vous êtes d'accord, il est probable que vous répondiez tous "oui". Cependant, si nous nous rencontrons et discutons, vous avez l'opportunité de partager votre propre pensée, et je peux également vous interroger. Chacun de nous interprétera les informations différemment, en mettant en lumière des aspects et des points de vue différents. En exposant nos pensées les uns aux autres, nous cherchons à instaurer une compréhension commune.

Le but réelle d'une user-story est la compréhension

Décomposer

L'User Story Map permet de décomposer les grosses user-stories au fur et à mesure qu'on l'a raconte. L'équipe va se rendre compte de problématique sous-jacent et cela va entraîner la création de nouvelles user-stories.

Créer

Écrivez des cartes ou des post-it pour extérioriser votre pensée lorsque vous racontez des histoires.

1. Rédiger quelques mots sur un post-it après avoir pensé à l'idée.
2. Expliquer son idée : mots, gestes, dessins. On raconte une histoire.
3. Placer la carte sur le mur.

Lorsqu'on raconte une histoire pas besoin de rédiger un roman, quelques notes, dessins, stickers pris en photo suffisent. Lorsqu'on regardera la carte nous nous remémorerons toutes les discussions qu'il y a eu autour d'elle.

La carte est un mémo sur les discussions menées et les accords passés.

Pour écrire une carte, vous pouvez suivre la méthodologie INVEST. A minima une carte on doit répondre aux questions :

- Pour qui : à qui s'adresse la fonctionnalité, quel utilisateur précisément.
- Quoi : qu'est-ce qui est attendu
- Pourquoi : fait-on cette fonctionnalité. Il faut donner du sens au travail à réaliser.

Organiser (Mapping)

Les user stories sont ensuite disposées sur une table (ou tableau) en fonction de leur relation les unes avec les autres et de leur importance. Cela crée une représentation visuelle qui met en évidence la séquence logique des fonctionnalités et permet de mieux comprendre la navigation et le flux d'utilisation du produit.

- Naturellement si on place une carte au-dessus d'une autre alors cela signifie qu'elle est plus importante.
- Naturellement si on positionne une carte à gauche puis une à droite cela signifie une séquence.

Cartographier nos stories nous aidera également à identifier les trous.

Pour en apprendre plus sur l'User Story Mapping vous pouvez vous procurer le livre User Story Mapping par Jeff Patton ou vous rendre sur son blog [5].

5. https://jpattonassociates.com/story-mapping/

Réunion quotidienne

Kanban

Dans le cadre de travail Kanban la réunion quotidienne se concentre sur le flux de travail et le mouvement des cartes dans le tableau. Le facilitateur Kanban et l'équipe mettent à jour le tableau, identifient les blocages, déterminent le travail le plus important à accomplir dans la journée.

Scrum

Avec Scrum, l'objectif de la réunion quotidienne est d'inspecter la progression vers l'Objectif de Sprint et d'adapter le Sprint Backlog si nécessaire, en ajustant les futurs travaux planifiés. De nombreuses équipes adoptent la démarche suivante.

```
Qu'ai-je fait hier ?
Que vais-je faire aujourd'hui ?
Est-ce que je vois un obstacle qui m'empêche d'avancer ?
```

Mais le focus doit être sur l'Objectif de Sprint, les questions deviennent donc

```
Qu'ai-je fait hier pour aider l'équipe de développement
à atteindre l'objectif du sprint ?

Que vais-je faire aujourd'hui pour aider l'équipe de
développement à atteindre l'objectif du sprint ?

Est-ce que je vois un obstacle qui m'empêche ou qui empêche
l'équipe de développement d'atteindre l'objectif du sprint ?
```

Definition of Done

La Definition of Done (DoD) est un ensemble d'éléments convenus qui doivent être achevés avant qu'une user-story soit considérée comme terminée. Elle sert à séparer les choses *en cours* des choses réellement *terminées*.

La Definition of Done c'est le fini de fini

Chaque organisation, chaque projet est libre de définir sa checklist de contrôle. Ci-dessous quelques éléments qu'on peut retrouver dans une Définition of Done :

- Le code est revu par les pairs
- Le code est déployé dans l'environnement de test
- Le code/la fonctionnalité passe les tests de régression
- Le code/la fonctionnalité passe les tests unitaires
- Le code est documenté
- La fonctionnalité est approuvée par les parties prenantes
- Les critères d'acceptation (de l'user-story) sont respectés

Une fois la tâche considérée comme finie (i.e. rempli la Definition of Done) alors elle est ajoutée à l'Incrément du Sprint.

Qui rédige la Definition of Done ?

La Definition of Done n'est pas imposée par une entité externe, elle est élaborée et approuvée par l'équipe Scrum pour créer de la transparence. Ceci permet à chacun de comprendre quel travail a accomplir et quelles normes doivent être respectées dans le cadre de l'incrément.

L'équipe s'engage à respecter la Definition of Done

Quand est rédigée la Definition of Done ?

La Definition of Done est créée avant le premier sprint de l'équipe et est ajustée lors de la rétrospective dans un but d'amélioration continue.

La Definition of Done n'est pas statique

Conséquences d'une absence de Definition of Done

L'absence de Definition of Done peut avoir des impacts majeurs sur la pérennité du projet :

- Augmentation de la dette technique
- Manque de confiance des parties prenantes
- Difficulté à mesurer le progrès ; qu'est-ce qui est réellement fini ?

Métriques

Introduction

Une équipe pourra s'améliorer uniquement si nous avons pu capturer et analyser l'effort passé. En Agile, l'accent est mis sur la transparence et l'inspection constante pour identifier les problèmes, réduire les gaspillages, et mettre en place des solutions pour améliorer les processus.

Il sera essentiel de faire preuve de prudence lors de l'analyse. Des métriques mal choisies ou mal interprétées peuvent conduire à des informations trompeuses ou à des comportements indésirables.

Erreurs néfastes

- Les métriques individuelles : Agile met l'accent sur le travail d'équipe, la collaboration et la responsabilité collective. Il devient inacceptable pour juger la performance individuelle. Ce comportement pourra nuire à la dynamique de l'équipe en incitant les membres à se connecter sur des objectifs personnels au détriment de l'objectif global de l'équipe.
- Le micromanagement : utiliser des métriques à des fins de surveillance de l'équipe conduira à un climat de méfiance envers management. Les individus seront moins enclins à entreprendre et à apporter un esprit critique.
- Ne pas consulter l'équipe : les membres de l'équipe Agile sont souvent les plus à même pour identifier les métriques pertinentes à leur situation. Si on remarque des variations inattendues alors des discussions *avec l'équipe* seront nécessaires pour comprendre les raisons et pour prendre les mesures appropriées.

Burndown Chart

L'objectif d'un tableau d'avancement (*Burndown chart*) est de fournir une transparence en temps réel sur l'avancement du projet, en aidant l'équipe de développement à visualiser facilement la progression, les potentiels problèmes et pour procéder à des ajustements.

Le burndown chart s'adresse à l'équipe. Il représente le reste à faire.

Interprétation

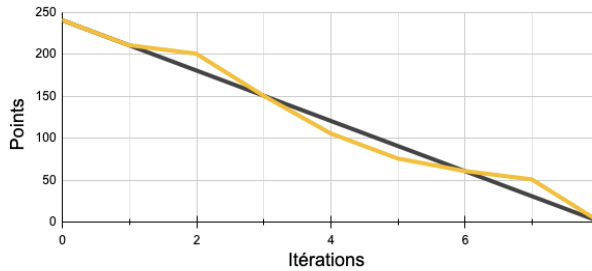


FIGURE 2 – Burndown Chart

Le Burndown Chart nous montre si l'avancement d'une release est sur la bonne voie. À la fin de chaque itération nous mettons à jour le graphique avec le nombre de stories points restants. Durant l'itération 1 nous avons accompli 40 story-points tandis que dans l'itération deux uniquement 10. Le Burndown Chart nous permet donc de visualiser facilement :

- Le taux d'avancement de la release
- La quantité de travail restante

Il faudra également veiller aux mauvaises interprétations. Pour exemple, une diminution de la vélocité de t ne veut pas forcément dire qu'une

tâche a été plus dure que prévu ou que l'équipe à eu des problèmes humains ou techniques. En effet ceci peut également signifier que le Product Owner à rajouter une tâche d'une valeur v. La vélocité se voit donc diminuer de t mais pour autant il n'y a eu aucun problème dans l'équipe.

Alternative Burndown Chart

Un graphique alternatif a été développé pour pallier à se manque. Il permet d'observer la variation de la vélocité **et** la variation du périmètre séparément.

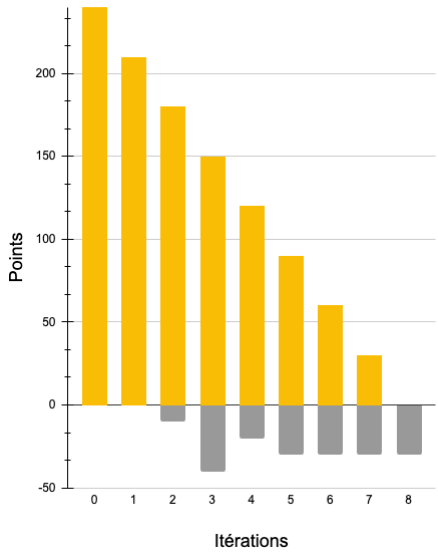


FIGURE 3 – Burndown Chart alternatif

La barre au-dessus de l'axe des abscisses correspond au nombre de points restant lors du début de l'itération. On remarque que l'équipe à une vélocité constante, mais que le Product Owner a identifié de nouvelles tâches, il les rajoute sous l'axe des abscisses. Au final l'équipe accomplit plus de travail qu'initialement prévu.

Il est important de respecter ces quatre règles :

- Chaque fois que le travail est achevé, le sommet est abaissé.

- Lorsque le travail est réestimé, le sommet est relevé ou abaissé.
- Lorsque de nouveaux travaux sont ajoutés, le bas est abaissé.
- Lorsque des travaux sont supprimés, le bas est relevé (au début de l'itération 4 le Product Owner à supprimer du travail)

Les dérives du Burndown Chart

Le Burndown Chart est un outil puissant pour visualiser facilement le reste à faire. Il n'est en aucun cas :

- Un outil pour suivre la vélocité individuelle, un tel comportement est à l'encontre de la culture Agile et pourra remettre en cause la réussite du projet. Entre terminer une user-story et aider quelqu'un, quelle incitation la vélocité individuelle va me donner ?
- Un outil pour le chef de projet. Le Burwndown Chart appartient à l'équipe, il permet d'anticiper les problèmes pour aider l'équipe à trouver une solution.

Burnup Chart

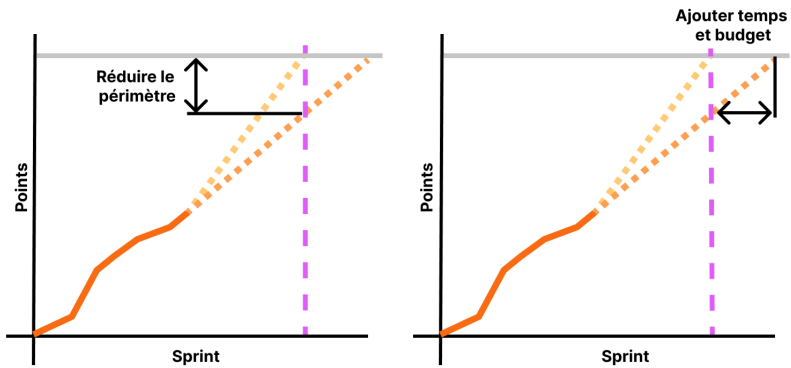
Le Burndup Chart permet de visualiser l'avancement du projet. En comparaison au Burndown Chart, qui lui, est réservée à l'équipe le Burnup Chart s'adresse en l'ensemble des parties prenantes.

Sources de variation

Le Burnup Chart met en évidence trois sources de variation :

- Les *variations du périmètre* dues à l'ajout et la suppression de fonctionnalités
- La *vélocité* de l'équipe par rapport à l'incertitude, la complexité des fonctionnalités, aux interruptions, etc ...
- Les *projections* qui illustrent des scénarios possibles en fonction de la vélocité

Interprétation et prédictions



En fonction de la vélocité de l'équipe et du périmètre, nous pouvons essayer de *prédire* quand le projet sera fini ou déterminer quoi adapter pour finir livrer à une date donnée. Dans la figure du dessus, la trajectoire idéale s'avère irréalisable (orange clair), nous devons donc nous adapter pour délivrer :

- Une première solution consiste à ajuste le périmètre afin de respecter deadline et budget
- Autrement, nous pouvons étendre le budget et la deadline afin que l'en- semble des tâches soient réalisées

Un Complément au Burndown Chart

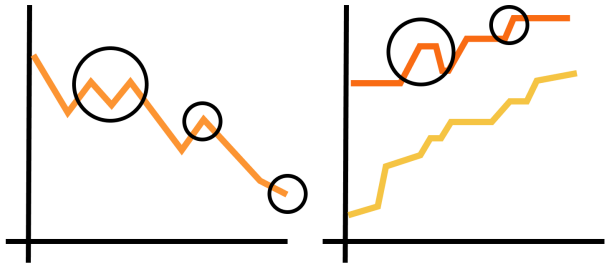


FIGURE 4 – Respectivement Burndown Chart et Burnup Chart

Dans la section précédente dédiée au Burndown Chart nous avions évoqué que malgré une vélocité de 20 au lieu de 30 sur une itération cela pouvait signifier que le PO avait rajouté du travail. Mais le Burndown Chart seul ne permettait pas de le savoir ; il fallait faire un Alternative Burndown Chart.

Maintenant si nous combinons le Burndown Chart et le Burnup Chart nous pouvons savoir si l'équipe a eu une vélocité inférieure ou si le périmètre a évolué. Dans la figure ci-dessus malgré une diminution de la vélocité à quatre reprises ceci s'explique à deux fois par une augmentation de la charge de travail (du périmètre).

Work In Progress (WIP)

Les notions de Work In Progress (WIP - travaux en cours) désignent le nombre de tâches ou d'éléments sur lesquels l'équipe travaille activement. Il comprend les tâches qui ont été commencées, mais qui ne sont pas encore achevées, ainsi que les tâches qui sont en cours et sur lesquelles on travaille activement.

En d'autres termes, le travail qui est "en cours".

Le Work In Progress est fortement lié à la notion de tableau kanban. Ainsi il peut être utilisé dans tous cadres de travail utilisant un tableau kanban (e.g Scrum comme Kanban). Le WIP est donc une mesure clé utilisée pour suivre et gérer le flux de travail dans le système. Des limites d'encours sont fixées dans les deux méthodologies pour aider à gérer le flux de travail et éviter de surcharger les membres de l'équipe.

Work In Progress en Scrum

Dans Scrum, le WIP est généralement estimé en comptant le nombre d'éléments du sprint backlog qui sont en cours de réalisation. Le sprint backlog représente la liste de tâches ou d'éléments que l'équipe s'est engagée à réaliser au cours du sprint actuel. L'équipe vient donc de définir sa limite maximale de WIP.

Work In Progress en Kanban

En Kanban, l'encours est généralement estimé en comptant le nombre d'éléments sur lesquels on travaille activement dans le tableau Kanban.

Limiter le Work In Progress

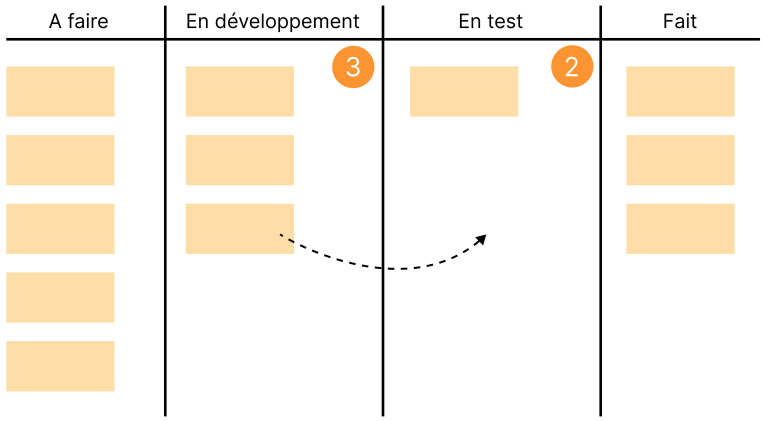


FIGURE 5 –

Quelque soit le système utilisé, l'équipe peut fixer une *limite d'encours* pour chaque étape du processus et ajuster les limites si nécessaire pour optimiser le flux de travail.

Dans l'illustration ci-dessus nous avons défini une limite de 3 éléments qui peuvent être en développement et 2 qui peuvent être en phase de test. Ainsi, si l'équipe de développement souhaite attaquer une nouvelle tâche, elle doit d'abord en finir une et l'envoyer pour les tests (WIP limité à 3).

- On évite la surcharge de travail et le multi-tâches
- On peut localiser les obstacles et réduire les goulets d'étranglement

L'application de limites d'encours nous permet de créer un flux de travail fluide et d'utiliser la capacité de travail de l'équipe à des niveaux optimaux.

Diagramme de flux cumulés

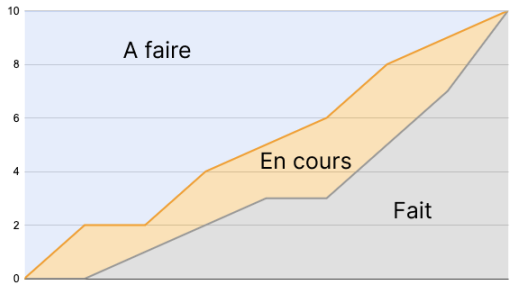
Un diagramme de flux cumulés (DFC) permet de visualiser l'état d'avancement du travail : quantité de travail en cours, en attente, effectué, etc ... Il donne des informations sur le rythme d'avancement du projet et nous aide à repérer les problèmes auxquels l'équipe a été confrontée.

Pour ce faire, sur l'axe horizontal on représente le temps, sur l'axe vertical de nombre de tâches et les courbes représentent de manière cumulative le nombre d'éléments dans un état :

- L'aire grise représente les tâches accomplies (croîs avec le temps)
- L'aire orange représente les tâches en cours de développement
- La dernière aire représente le nombre d'éléments dans notre backlog (décrois avec le temps)

Construction

La construction d'un Diagramme des flux cumulés se fait en reportant les informations d'un tableau kanban.



Jour	A faire	En cours	Fait
1	10	0	0
2	8	2	0
3	8	1	1
4	6	2	2
5	5	2	3
6	4	3	3
7	2	3	5
8	1	2	7
9	0	0	10

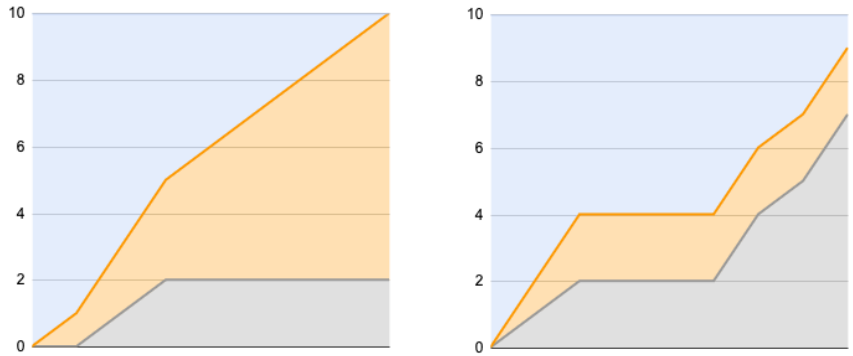
Le jour numéro 1 l'ensemble des tâches étaient dans la phase de développement. Le jour 5, 2 tâches étaient en développement et 3 tâches étaient réalisées. Et le jour 10, l'ensemble des tâches étaient réalisés.
Ainsi, les aires représentent le nombre d'éléments dans un état à un jour donné.

Lecture

A premier à bord on peut se demander à quoi nous sert un tel diagramme ? Pour comprendre son utilité, nous devons introduire quelques concepts de lecture.

Un goulot d'étranglement se traduit par un épaississement de la courbe ; l'équipe ne respecte pas le WIP

Un goulot d'étranglement se traduit par une barre qui stagne ; l'équipe n'avance pas



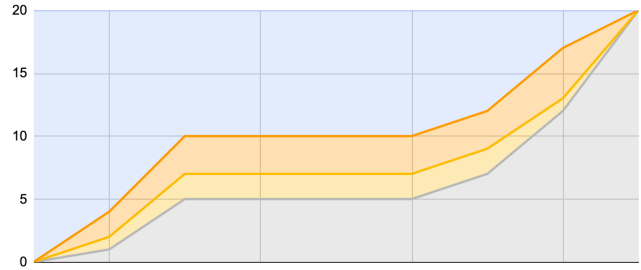
L'image de gauche traduit un non-respect du WIP, les développeurs prennent de nouveaux items sans finir les précédents. Est-ce que les développeurs sont bloqués sur une tâche et en commencent une autre ? Il faut faire une analyse. Dans l'image de droite, l'équipe a défini un WIP de 2, néanmoins pendant quelques jours aucune tâche n'a été terminée. L'équipe de développement a-t-elle eu une tâche plus compliquée qu'estimée ? Il faut faire une analyse.

Quelques patrons

Il existe plusieurs patrons facilement reconnaissables. Nous en présentons ici quatre avec les éventuelles causes. Afin de faciliter l'interprétation nous changeons les intitulés des aires : *Fait*, *En test*, *En développement* et *Backlog*

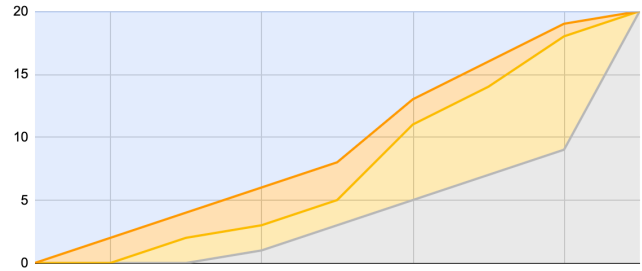
Aplatissement
Les causes possibles d'un aplatissement peuvent être :

- Problème technique où l'équipe prend du retard
- L'équipe est délocalisée sur un projet plus important
- La fonctionnalité a mal été évaluée, elle est plus longue que prévu



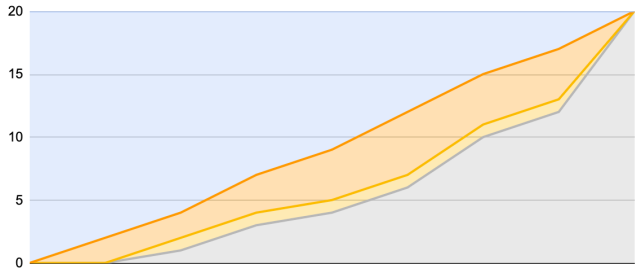
Augmentation
Les causes possibles d'une augmentation peuvent être :

- Un non-respect du WIP
- L'équipe a avancé sur le développement, mais elle s'approche de la fin du cycle. Ainsi elle augmente son effort sur les tests.



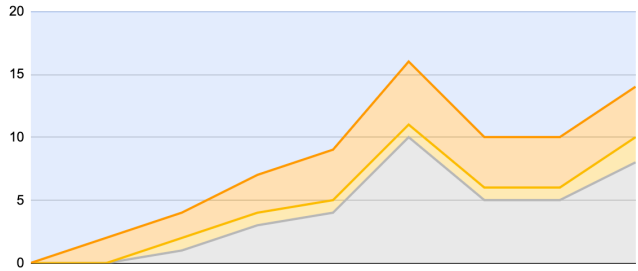
Rapprochement
Les causes possible d'un rapprochement peuvent être :

- L'équipe de test travaille sans difficulté
- L'équipe de développement a pris du retard et ne fournit plus les testeurs
- L'équipe de test est en manque de ressource et ne peut faire plus



Chute
Les causes possibles d'un rapprochement peuvent être :

- Des items auparavant acceptés ne le sont plus. L'équipe à construit la mauvaise chose



Le Diagramme de flux cumulés nous permet donc de nous questionner sur le fonctionnement du projet en visualisant les goulots d'étranglement. Plusieurs explications sont possibles pour un même graphique, il faudra donc établir une réunion avec l'équipe afin de récupérer les gênes pour de les éviter dans le futur.

Les dérives du Diagramme de flux cumulés

Lorsqu'on souhaite interpréter une Diagramme de flux cumulés il faut rester vigilant.

- Si le Diagramme de flux cumulés identifie des goulots d'étranglement, ils n'offrent pas toujours un contexte suffisant pour expliquer les raisons de ces goulots. Il faut donc veiller à réunir l'équipe pour comprendre le *pourquoi ?*
- Le Diagramme de flux cumulés nécessite de la rapidité et de l'exactitude dans la saisie des données. Les inexactitudes ou les retards dans la mise à jour des données peuvent fausser la représentation et conduire à des conclusions ou à des décisions incorrectes.

Escaped Defect

Un Escaped Defect (défaut échappé) est un défaut/bug qui n'a pas été découvert par l'équipe de test ou d'assurance qualité. Au lieu de cela, le défaut a été découvert par les clients.

Lorsque un problème est publié chez les clients, il est plus coûteux à plus corriger

Cette métrique compte simplement le nombre de défauts d'une release donnée qui ont été trouvés une fois celle-ci publiée.

Qui gère les défauts ?

C'est au Product Owner de déterminer comment traiter le défaut (e.g. le corriger, l'ignorer, etc). Il y a de nombreuses façons de gérer les bogues en production :

- Réserver du temps uniquement pour la dette technique et les corrections de bugs
- Le Product Owner ajoute ces tickets dans le backlog.

Les dérives

Si un bogue est arrivé en production, c'est qu'il n'a pas été détecté ; à qui est-ce la faute ?

- Si un développeur n'avait pas introduit le bogue, il n'aurait pas été mis en production.
- Si un testeur l'avait détecté, le bogue n'aurait pas été mis en production.
- Si un analyste avait rédigé de meilleurs critères d'acceptation, le bogue n'aurait pas été mis en production.
- Si le chef d'équipe avait encouragé une meilleure coopération et de meilleures réunions, le bogue n'aurait pas été mis en production.

Ne cherchez pas à trouver *un* coupable. Concentrez-vous sur la recherche de solutions pour prévenir la répétition de cette erreur.

- Tests incomplets
- Difficulté à prévoir toutes les interactions possibles
- Exigences mal comprises

Conclusion

Conclusion

Pour conclure, l'agilité n'est pas une boîte à outil. Lorsqu'une entreprise souhaite mettre en place cette démarche elle doit avoir conscience qu'elle ne gagnera pas plus d'argent, qu'elle n'ira plus vite mais qu'elle répondra au besoin du client avec une plus grande précision.

Pour ce faire, elle doit inculquer la philosophie Agile : mindset, valeurs et principes. L'agilité vise une optimisation globale du processus de création, des équipes pluridisciplinaires et une amélioration continue.

Une fois et seulement une fois que les objectifs de l'agilité sont compris l'équipe pourra réfléchit au framework à appliquer au quotidien. Les populaires sont Scrum et Kanban mais il en existe des dizaine d'autres (Nexus, LeSS, SAFe) permettant de s'adapter aux différents problèmes organisationnels (e.g. taille de l'équipe).

Finalement, en vue d'améliorer les performances de l'équipe, il est essentiel de recueillir et d'analyser les efforts. Différentes métriques sont disponibles, elles contribuent à mettre en lumière les difficultés rencontrées, mais leur interprétation doit être sujet à la prudence afin d'éviter toute dérive ou impact négatif.

Agile n'est pas un *boolean* mais un *float*. Ce n'est pas binaire, c'est un continuum. La question n'est pas de savoir si vous êtes "agile" ou "pas agile", mais de viser à être plus agile que vous l'étiez hier.