Agile

4 valeurs

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

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

12 principes

1. PRIORISER LA SATISFACTION DU CLIENT

Ceci est le principe le plus important des méthodes agiles. Cela peut paraître évident mais ça ne l'est pas forcément. En effet, pour savoir si le client est satisfait il faut savoir l'impliquer dans le processus de développement, parce qu'après tout, dans un cycle en V de deux ans on cherche aussi à satisfaire un client, mais il n'est pas ou peu impliqué.

Travailler en cycles courts ne suffit pas, même si vous avez désigné un Product Owner en interne pour valider les livraisons. Si ce n'est pas le client ou l'utilisateur final, cela ne fonctionnera pas.

2. ACCEPTER LES CHANGEMENTS

Ce principe est la définition même de l'agilité. Allégez les phases amont de conception et de définition du besoin, ceux-ci évolueront de toute façon au cours du projet, et organisez-vous pour accepter les changements sans que ce ne soit la panique à bord. Cela ne veut pas dire que l'on doive tout bouleverser à la moindre demande. Chaque imprévu doit être traité comme n'importe quelle autre fonctionnalité : on l'analyse, la chiffre, la priorise.

3. LIVRER EN PERMANENCE DES VERSIONS OPÉRATIONNELLES DE L'APPLICATION

Découper le cycle de développement en sprints ne suffit pas, il faut que chaque sprint soit couronné d'une ou plusieurs livraisons de features fonctionnelles. S'il vous faut plusieurs sprints pour faire une livraison valide, allongez la durée de vos sprints. Si au contraire pendant un sprint vous avez fait 15 mises en production et couvert 3 sujets, raccourcissez-les. Dans le premier cas le client croira que vous avez du retard, dans le deuxième il sera débordé par la quantité de fonctionnalités à valider.

4. ASSURER LE PLUS SOUVENT POSSIBLE UNE COOPÉRATION ENTRE L'ÉQUIPE DU PROJET ET LES GENS DU MÉTIER

Parlez-vous ! Oubliez les mails froids du type « Voici les specs pour le prochain sprint, on se revoit dans deux

semaines ». Ou le développeur qui dit à la fin du sprint « On a pas corrigé ce bug parce qu'on n'a pas réussi à la reproduire ». Echangez, clarifiez, demandez…

L'agencement des espaces de travail peut aider grandement quand métier et développement sont dans la même entreprise. Au lieu de regrouper tous les développeurs au même endroit (alors qu'ils ne travaillent pas forcément sur la même chose) et les gens du métier ailleurs, rassemblez les personnes en fonction des projets. Il faut toutefois éviter l'effet inverse : l'utilisateur qui va interpeller à tout bout de champ le premier développeur qu'il trouve pour lui remonter un bug ou faire une suggestion. Canalisez-les vers le Product Owner, il est là pour ça.

5. CONSTRUIRE LES PROJETS AUTOUR DE PERSONNES MOTIVÉES

Une équipe agile doit être motivée pour réussir. Si c'est une erreur de considérer les gens comme des ressources interchangeables, c'est d'autant plus vrai en méthode agile. En effet, une équipe agile a besoin de temps pour se « rôder ». En Scrum par exemple, les tâches sont chiffrées en points qui ne correspondent à aucune valeur. Il faut quelques sprints pour que l'équipe arrive à déterminer combien de points elle est capable de produire, parce que les membres apprennent à se connaître, apprennent des erreurs et des succès des sprints précédents. Transférer les membres d'une équipe à l'autre brutalement peut casser cet équilibre.

A lire sur le même sujet : La méthodologie Scrum : comment s'y prendre ?

6. FAVORISER LE DIALOGUE DIRECT

Parlez-vous ! Privilégiez l'oral, et limitez autant que possible l'écrit aux équipes éclatées géographiquement. Il n'y a rien de plus frustrant que de voir deux personnes dans la même pièce qui communiquent par Skype/Slack (sauf si c'est pour demander « Je peux venir te parler ?»).

7. MESURER L'AVANCEMENT DU PROJET EN FONCTION DE L'OPÉRATIONNALITÉ DU PRODUIT

Avec des méthodes telles que Scrum il est très facile de tout mesurer, de tout KPIser : points, burn-up/burn-down charts… C'est bien de produire de nombreux points mais cela ne suffit pas si ceux-ci reviennent sous la forme de bugs dans un ou deux sprints. Le but de chaque itération est de produire du logiciel qui fonctionne, idéalement dans les temps estimés, c'est la meilleure façon d'évaluer la performance d'une équipe.

8. ADOPTER UN RYTHME CONSTANT ET SOUTENABLE PAR TOUS LES INTERVENANTS DU PROJET

Si les méthodes agiles doivent vous faire tendre vers une plus grande adaptabilité, elles ne sont pas synonymes de chaos pour autant. Etre agile, ce n'est pas tout interrompre suite à une demande du client, faire travailler l'équipe toute la nuit et se féliciter ensuite d'avoir été réactif, c'est même tout le contraire ! Si une demande de changement doit provoquer de la friction et amener l'équipe à l'épuisement, c'est justement le signe que vous n'êtes pas agile. N'oubliez pas également que le client/Product Owner fait partie de l'équipe : produire des quantités de fonctionnalités que le PO n'a pas le temps de valider n'est pas un rythme soutenable.

9. CONTRÔLER CONTINUELLEMENT L'EXCELLENCE DE LA CONCEPTION ET LA BONNE QUALITÉ TECHNIQUE

Agile ne doit pas vous faire abandonner la conception, sinon vous allez essayer d'assembler des bouts de code produits par différentes personnes et espérer que tout fonctionne. Ce qui est à bannir, c'est la phase de conception sur toutes les fonctionnalités du projet en amont. En tant que Scrum Master, demandez à vos équipes

de faire de la conception avant de coder. En tant que Product Owner, demandez leurs d'écrire des specs détaillées pour s'assurer que l'histoire a été comprise. Le but n'est pas de produire de la documentation, cela peut être une liste dans un mail, un bout de pseudo-code ou même une discussion. Si vous travaillez en Scrum ou en Kanban, vous pouvez rajouter une colonne Conception sur le tableau, entre To Do et On Going. Toutes les cartes ne seront pas concernées. Un bug qui se corrige en une heure n'en aura pas forcément besoin tandis qu'une carte qui prendra une journée et plus devrait passer systématiquement par de la conception.

10. PRIVILÉGIER LA SIMPLICITÉ EN ÉVITANT LE TRAVAIL INUTILE

Simplifiez au maximum. Si le projet ressemble à une montagne insurmontable, ne visez pas la fin mais seulement la première étape. Au besoin, divisez chaque étape autant qu'il le faudra pour que chaque tâche, chaque histoire paraisse simple.

11. AUTO-ORGANISER ET RESPONSABILISER LES ÉQUIPES

Vous aurez beau avoir les meilleurs talents, une équipe travaillant sous la contrainte sera toujours moins performante, laissez-les s'auto-organiser. Les personnes qui travaillent vers un but commun parce qu'elles le veulent seront toujours plus efficaces et plus fiables que des équipes travaillant ensemble parce qu'on leur a dit de le faire.

12. AMÉLIORER RÉGULIÈREMENT L'EFFICACITÉ DE L'ÉQUIPE EN AJUSTANT SON COMPORTEMENT

L'amélioration continue est un principe que l'on doit garder en tête quand on travaille en méthode agile. Pourtant, la rétrospective est généralement la réunion qui passe à la trappe : c'est à la fin du sprint, souvent le vendredi, les gens sont fatigués, n'en voient pas l'utilité… C'est une erreur car il n'est pas possible de s'améliorer en arrière. Ne négligez pas ce moment-là, c'est peut-être le plus enrichissant pour vos équipes.

Nous venons de voir pourquoi il est important d'avoir ces principes toujours à l'esprit, ils vous permettent de garantir l'agilité de vos équipes quelle que soit la méthode que vous employez. De plus, grâce à ces fondamentaux, vous pouvez même implémenter votre propre méthode agile !

User stories

Une User Story est une description simple et compréhensible d'une fonctionnalité. Mais ayez en tête qu'une User Story, comme son nom l'indique, est avant tout une histoire qui se raconte, créé la discussion et amène l'équipe à confirmer sa compréhension du besoin.

Une user story se compose de :

- D'un titre explicite :
  Par exemple : « Client détenteur d'une carte VISA règle sa commande ».
- D'une phrase narrative structurée sous la forme « En tant que … Je veux … Afin de … ».
  Par exemple : « En tant que client détenteur d'une carte VISA, je veux saisir mes données bancaires afin de régler ma commande en ligne avec ma carte VISA ».
  Cette formulation permet au Product Owner d'apporter une vision orientée client et d'identifier précisément la fonctionnalité et le bénéfice attendu.
- D'un ensemble d'exigences et de critères d'acceptation :
  Par exemple : «Contrôle à effectuer sur le format de carte ».

Une fois affinée, une User Story a une valeur business ou valeur métier. La valeur métier est décidée suite à une demande de priorisation du Product Owner à son client. Elle permet ainsi au client de s'engager sur une priorité, et au Product Owner d'indiquer la priorité métier à son équipe dans le backlog de produit.

La forme canonique d'une User Story

Une User Story présente une vision utilisateur autour de 3 axes : rôle, besoin et valeur métier

`<En tant que>` rôle, utilisateur
`<Je Veux>` besoin, action
`<Afin de>` bénéfice, valeur métier

Pour bien découper les types d'utilisateurs d'une user story, posez-vous la question : « De qui ai-je besoin d'obtenir le feedback sur la fonctionnalité ?", pour savoir si la user story résout réellement son problème ou encore "Si cette fonctionnalité n'est pas développée, qui va en pâtir ?".

Affiner le "afin de" permet de clarifier la valeur apportée par la user story, et donc d'aider à la priorisation des user stories des prochains sprints. Si vous avez des difficultés à exprimer correctement l'objectif de la user story (le "afin de"), raisonnez en négatif : "que se passe-t-il si on ne le fait pas ? Quels sont les risques?"

Cette approche de rédaction est donc guidée par de bonnes questions à se poser. En voici 3 simples à se poser lors de la rédaction de vos User Stories :

- Qui a fait la demande ou qui bénéficie de la demande ? -> le rôle utilisateur
- Quelle est la demande ? -> le besoin
- Quelle valeur métier découle de la réalisation de ce besoin ? -> le bénéfice

`<En tant que>` rôle, utilisateur - Qui a fait la demande ?
`<Je veux>` besoin, action - Quelle est la demande ?
`<Afin de>` bénéfice - Quelle valeur métier découle de la réalisation de ce besoin ?

Une “User Story” ça veut bien dire “histoire utilisateur”, pas “contrainte technique”

Dans le cas de mes projets, portant sur des SI commerciaux et de gestion, il est facile de tomber dans des cas de rédaction du type "En tant que `<SI partenaire>`, je veux consommer les données X, Y et Z afin de vérifier la véracité de mes données avec celles du SI appelé...".
Si c'est le cas, reprenez les questions :

- Qui a fait la demande ?
- Quelle est la demande ?
- Quelle valeur métier découle de la réalisation de ce besoin ?

... car vous voulez éviter ce type de User Stories bien trop génériques.

Une bonne User Story respecte les caractéristiques réunies sous le sigle INVEST.

Si l'on suit ce framework INVEST, une bonne User Story est :

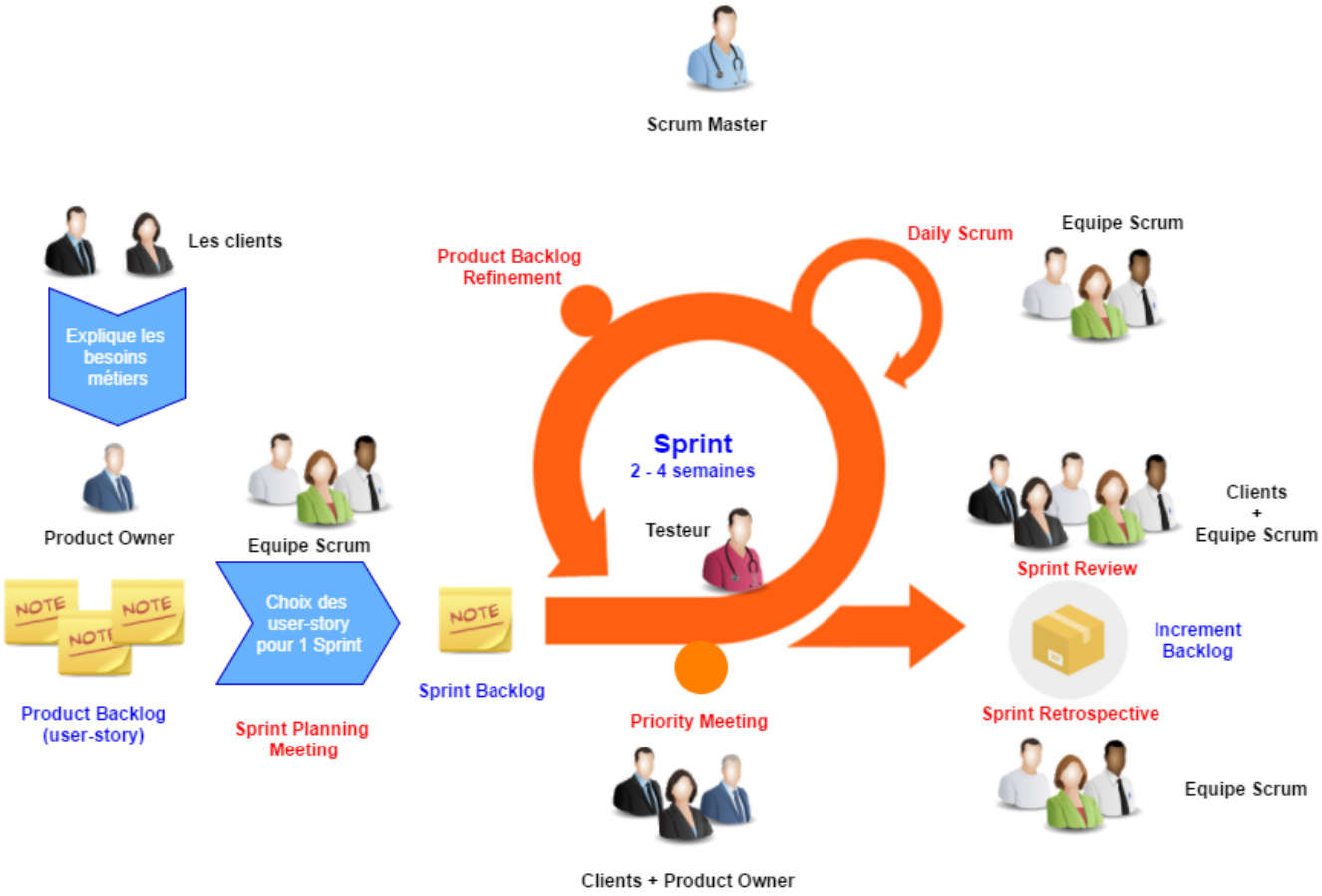
- Indépendante : une User Story est indépendante vis-à-vis des autres User Stories du backlog. Idéalement, elle se suffit à elle même pour éviter les dépendances avec d'autres User Stories. Car toute dépendance génère des problématiques de planification et de tests.

- Négociable : une User Story est un support de discussion en vue d'une amélioration du besoin initial. Elle peut être modifiée, on parle d'affinage jusqu'à son intégration dans une itération ou Sprint. Généralement,

on considère qu'elle est affinée lorsqu'elle répond aux critères de la "Définition of Ready" établie par l'équipe.

- Valorisable : la réalisation d'une User Story doit rendre un service à l'utilisateur. Elle n'a de sens que si elle apporte une valeur métier.
- Estimable : une User Story doit être bien définie pour être facilement estimable, quel que soit votre mode d'estimation. Si l'équipe n'est pas en capacité de l'estimer, c'est souvent que votre User Story manque de clarté.
- Small : une User Story doit être réalisable sur un sprint, soit suffisamment petite ou découpée de telle manière à pouvoir être déployée sur un seul sprint et minimiser les effets tunnels sur plusieurs sprints. Cherchez donc à découper vos User Stories le plus finement possible. Mieux vaut créer deux petites User Stories, qu'une grande.
- Testable : une User Story doit raconter une histoire dont les critères d'acceptabilité et les tests doivent découler de manière évidente pour faciliter sa validation.

Les rituels Scrum



Le sprint planning

- **L'objectif** : bien organiser le sprint scrum qui démarre.
- **La durée** : le temps qu'il faut pour finir la préparation du sprint. Mais c'est préférable de ne pas dépasser plus de 2h.
- **Les participants** : le Product Owner (PO), le Scrum Master (SM) et toute l'équipe de développement (des développeurs)
- **Les animateurs** : c'est Le Product Owner qui est le maître de cérémonie. Le Scrum Master sera plutôt

celui qui va seconder le Product Owner pour intervenir en cas de besoin.

Le déroulement :

1. Le Product Owner définit les objectifs du sprint. N'hésitez pas à rappeler ces objectifs tout au long de la Sprint Planning voire les écrire directement sur le board.

2. L'équipe va devoir ré-estimer le point d'effort (la complexité) de chacune des demandes du sprint précédent qui ont fini en « work in progress » (en cours).
   Ce travail est inutile sur les demandes qui étaient en « todo » à la fin du précédent sprint.

3. Le Product Owner va alors proposer aux développeurs un ensemble de demandes qu'ils devront prendre en charge pendant le sprint. Ces demandes répondront dans l'ensemble aux objectifs fixés en début de cérémonie.

4. On n'estime plus des demandes pendant le sprint planning mais on réalise les estimations en amont du Sprint Planning. C'est lors de la Product Backlog Refinement que l'estimation est faites. Elle permet à la sprint planning d'être plus rapide et de fluidifier au maximum les processus. En revanche, l'équipe peut estimer exceptionnellement une demande du backlog indispensable à traiter lors de ce sprint malgré le fait qu'elle ne soit pas passé en Product Backlog Refinement.

5. La dernière étape est de découper l'ensemble des demandes en sous-tâches techniques. Il n'y a pas de règles précises pour ce découpage technique ; je conseille aux équipes techniques de faire le découpage avec lequel elle se sentira le plus à l'aise possible.

Chaque jour du sprint scrum (sauf le jour du sprint planning), l'équipe participe au daily scrum.

Le daily scrum

- **L'objectif** : l'équipe de développement doit être claire sur l'avancement du projet et relever des points bloquants.
- **La durée** : 15 minutes maximum tous les jours du sprint scrum
- **Les participants** : l'équipe des développeurs, le Product Owner (en observateur toléré) et le Scrum Master (nécessaire si l'équipe n'est pas mature)
- **Les animateurs** : l'équipe de développement autonome

Le déroulement :

On fait un tour de l'équipe de développement et chaque participant doit :

- Expliquer ce qu'il a fait depuis la dernière Daily
- Dire ce qu'il pense faire jusqu'à la prochaine Daily
- Lever une alerte (s'il y en a)

Attention : chaque membre n'a que 2 minutes pour parler. Si un problème est soulevé et qu'on va probablement passer du temps à en parler, le Scrum Master devra dès le début proposer de parler de ce problème après la Daily afin de ne pas perturber celle-ci.

En cours du sprint, une fois par semaine, l'équipe participe au Product Backlog Refinement anciennement appelé Grooming.

Le Product backlog Refinement

- **L'objectif** : affiner le contenu du Product Backlog et bien organiser les prochains Sprint.

- **La durée** : ne dure pas plus d'une heure par session, elle peut également durer que dix minutes si le Product Owner n'a pas besoin de plus de temps
- **Les participants** : l'équipe de développement, le Product Owner et le Scrum Master
- **Les animateurs** : C'est Le Product Owner qui est le maître de cérémonie. Le Scrum Master sera plutôt celui qui va seconder le Product Owner pour intervenir en cas de besoin

Le déroulement :

1. Le Product Owner va proposer des user-stories aux développeurs. Pour chaque demande les développeurs posent des questions au Product Owner pour affiner la demande et dans le but de pouvoir l'estimer si la demande n'est pas 100% claire.
   Si ils ont totalement compris la demande, ils valident le ticket et l'estimeront
   Sinon, si les développeurs ne comprennent pas le ticket, cela imposera au Product Owner de revoir sa copie jusqu'à la prochaine session.
2. Les développeurs vont alors faire une estimation de la demande ensemble en temps ou en point d'effort (fortement recommandé).

Le sprint scrum se termine par la sprint review qui doit avoir lieu juste avant la sprint retrospective.

La sprint review

- **L'objectif** : faire un point sur ce qui a été réalisé en cours de sprint. Il est aussi intéressant de présenter le rendu afin d'améliorer l'échange entre l'équipe et les parties prenantes, cela dans le but de récupérer un maximum de feedback constructifs.
- **La durée** : ne dure pas plus de 1 heure
- **Les participants** : l'équipe de développement, le Product Owner, le Scrum Master, les utilisateurs clés [optionnel], les parties prenantes [optionnel] et les sponsors [optionnel]
- **Les animateurs** : C'est Le Product Owner (PO) qui est le maître de cérémonie, Le Scrum Master (SM) sera le second animateur. C'est plutôt conseillé que le PO et le SM présentent en binôme.

Le déroulement :

1. Le Product Owner présente le travail parcourus pendant le Sprint
2. Des développeurs font une démonstration d'une nouvelle fonctionnalité terminée ou d'un nouveau produit terminé
3. Le Scrum Master présente quelques indicateurs révélateurs (burndown chart, burnup chart, impediment donut…) pour expliquer concrètement l'avancement du projet.
4. Questions/Réponses et feedback.

La sprint retrospective

- **L'objectif** : de travailler sur deux axes principaux soit l'amélioration continue et la santé de l'équipe
- **La durée** : une heure
- **Les participants** : l'équipe de développement, le Product Owner et le Scrum Master
- **Les animateurs** : C'est le Scrum Master qui prépare et anime à 100%

Le déroulement :

Le scrum Master va proposer des ateliers ludiques afin d'atteindre des objectifs de la rétrospective avec quelques règles d'or :

chaque rétrospective doit être différente
elle doit être animée et pas monotone