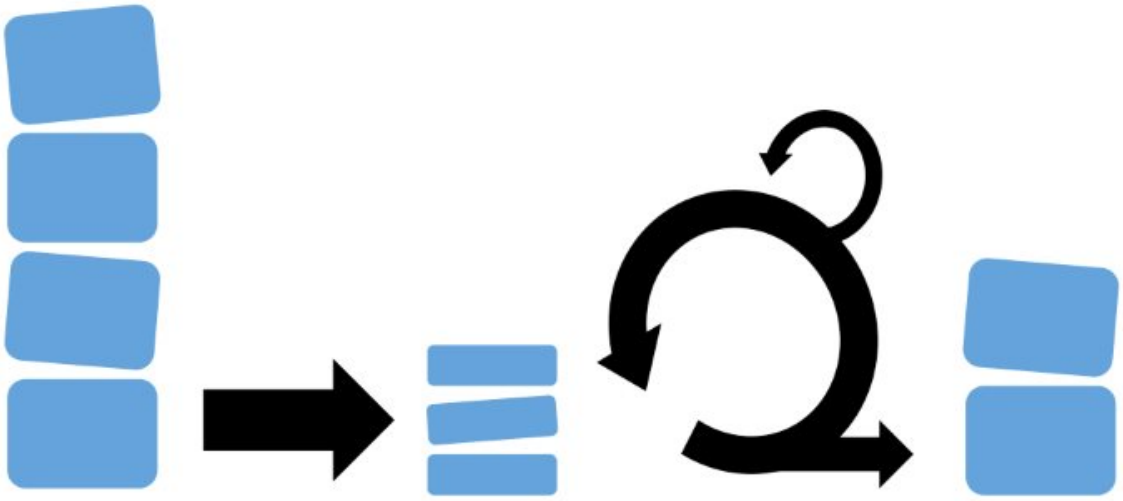
GUIDE DE DÉMARRAGE SCRUM

Faire le bon produit au bon moment et au meilleur coût



FLORENT LOTHON

Expert en gestion de projet et management d'équipe

L'Agiliste

Motivation et performance durables

SOMMAIRE

Introduction

La méthode Scrum (« Scrum » signifie « Mêlée » en anglais), ou plus exactement le cadre méthodologique Scrum est de loin la méthode Agile la plus utilisée dans le monde. Expérimentée en 1993, elle bénéficie aujourd'hui de nombreux retours d'expérience. Les conférences, communautés, formations, blogs, outils et ouvrages à son sujet ne manquent pas.

L'objectif de cet article est de vous aider à vous lancer dans la mise en oeuvre de Scrum. Il décrit le processus associé, ses étapes, réunions, rôles, etc. Dès que les limites de cet article seront atteintes, la lecture du Guide Scrum (20 pages, gratuit, complet et traduit en français) puis du livre « Scrum et XP depuis les tranchées » (160 pages, également gratuit, complet et traduit en français) est recommandée. Pour une meilleure compréhension de cet article, il est préférable d'avoir assimilé les principes Agile. Au besoin, je vous invite à lire la fiche pratique « Introduction aux méthodes Agile ».

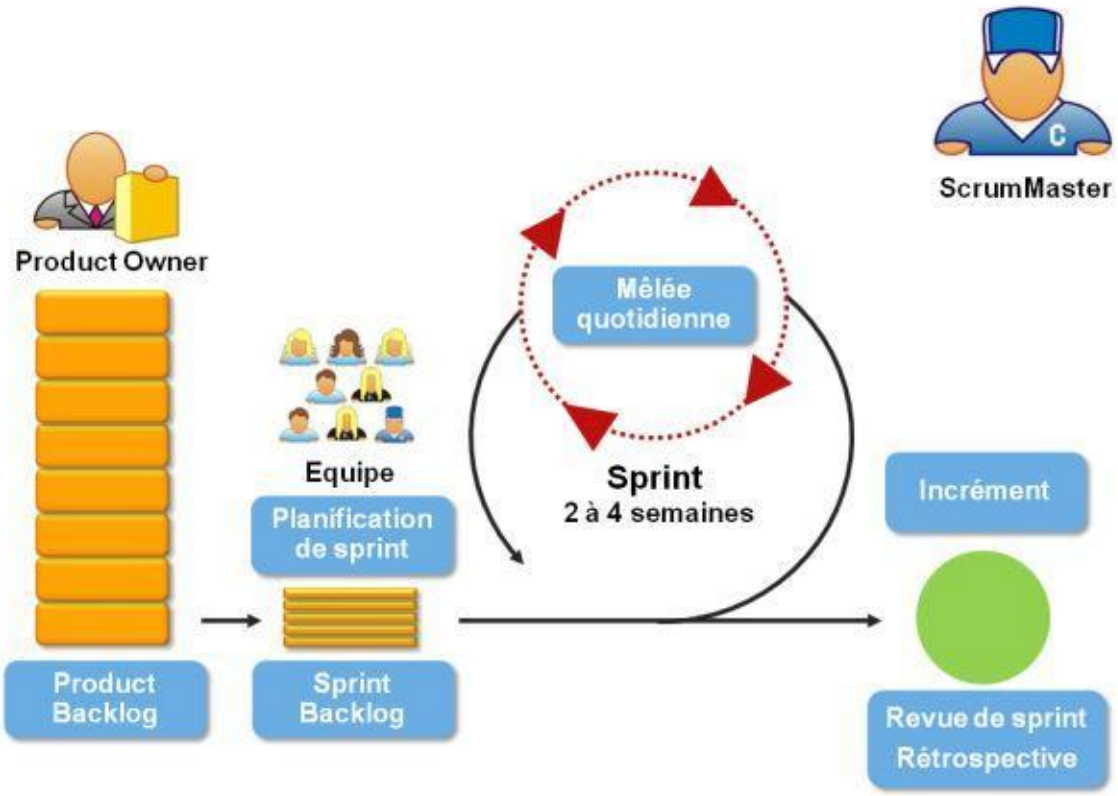
Au sujet de Scrum

Parler d'une « méthode » concernant Scrum n'est pas ce qu'il y a de plus approprié. Scrum ne se considère pas comme une méthode mais comme un cadre méthodologique. Une méthode dit généralement « comment » faire les choses. Scrum ne dit pas comment réussir son logiciel, comment surmonter les obstacles, comment développer, comment spécifier, etc. Il se contente d'offrir un cadre de gestion de projet Agile (et c'est déjà beaucoup) : des rôles, un rythme itératif, des réunions précises et limitées dans le temps, des artefacts (product backlog, sprint backlog, graphique d'avancement) et des règles du jeu.

Au sein de ce cadre méthodologique de gestion de projet, les acteurs ajustent empiriquement, au fil des itérations, leur propre méthode en fonction de leur contexte. On peut qualifier Scrum de simple, pragmatique, transparent et empirique. Scrum ne couvrant que les aspects de gestion de projet, c'est souvent la méthode eXtreme Programming (XP) qui vient compléter le vide laissé en matière de pratiques de développement. XP apporte ainsi les pratiques de programmation en binôme, de développement piloté par les tests (TDD ou Test Driven Development), intégration continue, etc. Ces dernières jouent un rôle capital pour relever le défi du développement incrémental. Le mouvement Software Craftsmanship est également là pour répondre à ces enjeux techniques.

NB : Sachez que eXtreme Programming couvre également efficacement les aspects de gestion de projet, faisant d'elle l'une des méthodes Agile les plus complète qui existe.

Utilisation de Scrum



Processus Scrum (source des icônes des personnages : Mike Cohn)

Pré requis recommandés

- Un grand mur libre et dégagé dans l'espace de travail de l'équipe.
- Blocs de post-it et marqueurs.
- L'ouvrage « Scrum et XP depuis les tranchées ».
- Jeu de cartes ou logiciel de Planning Poker.

Les Rôles en bref

Scrum définit seulement 3 rôles :

- Le **Product Owner** qui porte la vision du produit à réaliser et travaille en interaction avec l'équipe de développement. Il s'agit généralement d'un expert du domaine métier du projet.
- L'**Équipe de Développement** qui est chargée de transformer les besoins exprimés par le Product Owner en fonctionnalités utilisables. Elle est pluridisciplinaire et peut donc encapsuler d'autres rôles tels que développeur, architecte logiciel, DBA, analyste fonctionnel, graphiste/ergonome, ingénieur système.

- Le **Scrum Master** qui doit maîtriser Scrum et s'assurer que ce dernier est correctement appliqué. Il a donc un rôle de coach à la fois auprès du Product Owner et auprès de l'équipe de développement. Il doit donc faire preuve de pédagogie. Il est également chargé de s'assurer que l'équipe de développement est pleinement productive. Généralement le candidat tout trouvé au rôle de Scrum Master est le chef de projet. Celui ci devra cependant renoncer au style de management « commander et contrôler » pour adopter un mode de management participatif.

Vision du produit et product backlog

La première étape consiste à formaliser la vision du produit (logiciel) que l'on souhaite réaliser. Cette vision décrit les principaux objectifs, jalons, utilisateurs visés. Elle contribuera à guider et fédérer les acteurs du projet. La suite consiste à établir la liste des exigences fonctionnelles et non fonctionnelles du produit. Chaque exigence est ensuite estimée par l'équipe de développement avec la technique de Planning Poker. A la lueur des estimations, la liste ainsi complétée est ordonnancée. Les exigences seront converties en fonctionnalités utilisables selon cet ordonnancement. Le principe étant de convertir en premier les exigences qui apportent le plus de valeur ajoutée (ou ROI) au commanditaire. Il s'agit donc de faire remonter les exigences fonctionnelles de la plus haute valeur ajoutée (ou dont le ROI est le plus élevé) en haut de la liste. Cette liste est appelée le Product Backlog. Le Product Backlog servira à piloter l'équipe de développement et pourra évoluer tout au long du projet. Le changement est non seulement autorisé mais encouragé afin de pouvoir éliminer les idées de départ qui s'avéreront mauvaises et de prendre en compte les nouvelles idées qui arriveront en cours de route. Cette activité de construction du Product Backlog est collaborative, elle implique le Product Owner et l'équipe de développement.

ET DANS LE CAS D'UN APPEL D'OFFRE ?
Vous répondez à un appel d'offre pour la réalisation du logiciel de votre client ? Vous ne pouvez pas construire le Product Backlog avec lui ? Dans ce cas, vous pouvez partir du cahier des charges, extraire de ce dernier les exigences, les estimer (idéalement avec 3 membres de l'Equipe de Développement pressentie) et initialiser ainsi le Product Backlog. Vous pouvez également aller plus loin en proposant un premier ordonnancement.

Pondérer chaque exigence d'une valeur ajoutée n'est pas toujours évident pour une équipe métier (MOA) novice. Différentes techniques s'offrent à vous. Le fameux Planning Poker (voir paragraphe ci dessous) peut s'avérer utile pour déterminer collectivement les pondérations en « points » de valeur ajoutée. On peut également utiliser des échelles de valeur plus ou moins fine (exemple : « faible », « moyenne », « haute » ou encore les tailles de tee-shirt : XS, S, M, L, XL, XXL).

Estimations des exigences

L'ensemble des fonctionnalités de la liste doivent être estimées par l'équipe de développement afin de permettre les futurs engagements de cette dernière. Impliquant plusieurs développeurs, le Planning Poker permet de mettre à profit les expériences de chacun et de parvenir rapidement à une estimation optimale et objective. Avant ou pendant les estimations, le Product Owner pourra être sollicité afin de répondre aux questions de l'équipe de développement. A ce stade, le besoin pourra être approfondi, mais sans aller trop loin

(il s'agit simplement d'estimer le coût de chaque exigence). La conception détaillée se fera pendant les itérations (sprints).

Lien pour en savoir plus sur le Planning Poker

Démarrage

Vous pouvez commencer par déterminer ensemble (équipe de développement et product owner) la durée des itérations ou Sprints (4 semaines maximum). Cette durée devra être la même pour l'ensemble des sprints afin de maintenir un rythme régulier propice aux automatismes et pouvoir construire des indicateurs de pilotage fiables.

Un projet démarre généralement par les travaux préparatoires du projet tels que la construction du **product backlog** et de la **vision du produit**, initiation des acteurs à Scrum (sur un projet de développement logiciel, on peut étendre à la préparation des environnements, mise en place de l'intégration continue, définition de l'architecture générale du projet, etc.). Inutile de le faire durer trop longtemps cette phase, souvenez vous (cf. article « introduction aux méthodes Agile »), l'idée est de se lancer sans élaborer au préalable un plan et une architecture millimétrés qui risqueraient de nous enfermer, de nous frustrer, voire de nous coûter cher à courts et longs termes. L'architecture doit être souple et émerger au fil des sprints.

DISPARITION DE LA NOTION DE "SPRINT 0"
Il fut un temps où cette étape préparatoire ou étape de cadrage s'appelait le "Sprint 0" (y compris dans le guide officiel Scrum de l'époque). Étant entendu que ce sprint 0 pouvait avoir une durée différente des sprints "normaux" qui suivent. Malheureusement, beaucoup d'équipes ont trouvé en ce sprint 0 l'opportunité d'en faire une phase de "conception" et retombaient dans le piège de l'approche traditionnelle consistant à tout anticiper. Cette étape n'est absolument pas une phase de conception/planification détaillée. Vous voilà donc averti 😉

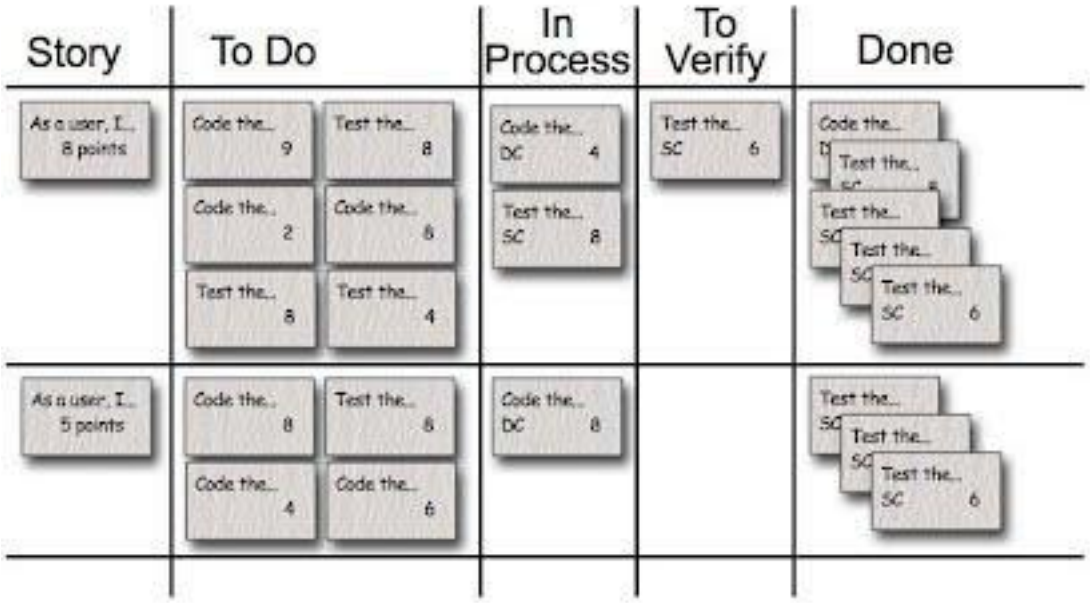
Réunion de planification de sprint

Durée maximum : 2 heures par semaine de sprint (autrement dit : 4 heures pour des sprints de 2 semaines).

Phase 1 : Le « Quoi »

Une fois que le Product Backlog est suffisamment complet et ordonnancé, on peut planifier un sprint. LeProduct Owner revoit alors avec l'équipe de développement la vision du produit, la roadmap, le plan de livraison (jalons et deadline), l'objectif du sprint et le Product Backlog. L'équipe de développement vérifie les estimations, confirme qu'elles sont exactes et sélectionne en haut du Product Backlog les exigences qu'elle se sent capable de convertir en fonctionnalités utilisables d'ici la fin du sprint (il s'agit d'une prévision et non pas d'un engagement « contractuel »).

Phase 2 : Le « Comment »



Exemple de tableau des tâches

L'équipe de développement fait ensuite l'inventaire des tâches qui permettront de convertir les exigences sélectionnées en fonctionnalités utilisables d'ici la fin du sprint. Toutes les exigences n'ont pas nécessairement besoin d'être découpées en tâches. En cas de manque de temps, l'équipe de développement peut se contenter de découper celles qui seront réalisées au cours des premiers jours du sprint (elle découpera en cours de sprint les autres exigences). Elle doit cependant aller suffisamment loin dans l'effort de conception pour pouvoir vérifier sa prévision. Si elle constate après analyse des exigences sélectionnées, que sa prévision est erronée, elle peut réajuster avec le Product Owner la liste des exigences sélectionnées.

Les tâches de développement sont centralisées dans le Sprint Backlog et ajoutées au tableau des tâchesphysique (aussi appelé Kanban, même si Kanban veut dire bien plus). L'idéal est de parvenir à un découpage relativement fin (inférieur ou égal à une journée de travail).

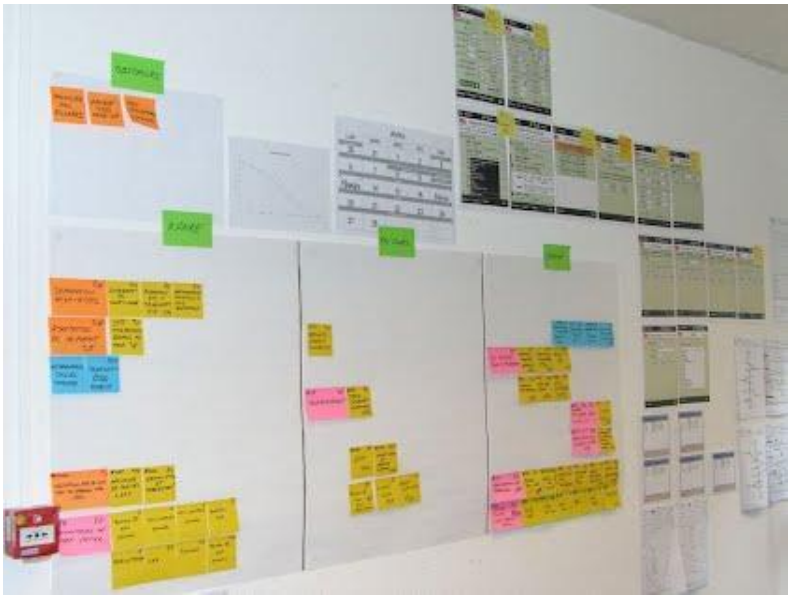


Photo d'un tableau des tâches

Chacun peut personnaliser les colonnes de son tableau des tâches, ou code couleur des post-it. A titre d'exemple, voici la photo du tableau que nous avons utilisé sur l'un de nos premiers projets (au passage, vous pouvez également jeter un œil à la vidéo tournée sur ce projet et au retour d'expérience associé). Les User Stories (une User Story représente une exigence, voir annexe pour plus de détails) sont les gros post-it larges (rose et orange), les sous tâches des User Stories sont en jaune, on trouve en bleu des tâches imprévues indépendantes. Au dessus du tableau figurent les obstacles courants, un graphique d'avancement appelé burdown chart ainsi qu'un calendrier géant avec les dates clefs et les congés de chacun. Sur la droite on trouve des maquettes d'IHM, des documents métier, etc. Cette réunion de planification est l'occasion de préciser ou rappeler à l'équipe la définition de « terminé » pour une user story ou exigence. Exemple de définition de « terminé » : code commité, testé unitairement, documenté, testé en intégration, revue par un pair, tests d'acceptation de la user story passants.

Sprint (2 à 4 semaines)

Au cours du Sprint, l'équipe se concentre sur l'accomplissement des tâches du Sprint Backlog. En cas de retard (indiqué par le Burndown Chart), des exigences ou tâches seront retirées du Sprint Backlog en cours de route en essayant de préserver l'objectif du sprint (pour cela, il est conseillé d'ordonnancer les exigences au sein du sprint). Et inversement, si l'équipe avance plus vite que prévu, des exigences ou tâches y seront ajoutées. En accord avec le Product Owner dans les deux cas.

Les développements se font verticalement et non pas horizontalement par couche. Le but est de développer les fonctionnalités de bout en bout (de la conception aux tests) au fil de l'eau au cours du sprint. Autrement dit d'éviter un mini cycle en V au sein du sprint, voire de se retrouver avec une surcharge d'effort de test en fin de sprint. Les développeurs doivent donc éviter de trop paralléliser les exigences et encore moins les tâches de développement. Pour cela, le pair programming peut se révéler utile ainsi que la définition d'une limite maximum d'éléments au sein d'une colonne du tableau des tâches (voir notion de Work In Progress du Kanban).

Si l'équipe de développement n'est pas convaincue, le « jeu du prénom en mode multitâche » peut s'avérer utile.

Le tableau des tâches physique rempli de post-it est pratiquement indispensable. Il permet d'avoir une vision claire du travail à accomplir, en cours et terminé. Il peut également s'avérer précieux lors des réunions quotidiennes (voir § suivant), surtout si vous calculez à la main le « reste à faire » du Sprint afin de tracer le graphique d'avancement (voir plus loin le « Burndown Chart »). Le tableau facilite également l'affectation des tâches par l'équipe en ayant une vision d'ensemble du sprint en un coup d'œil. Inutile de se torturer à planifier à l'avance dans le détail l'activité de chaque développeur sur toute la durée du sprint. Il faut se détacher d'une approche prédictive et des diagrammes de Gantt et renoncer au style de management « commander et contrôler ». Ce sont les développeurs qui « tirent » les tâches et non pas le Scrum Master qui les affecte.
Pendant le sprint, l'équipe de développement assistera le Product Owner dans ses activités d'affinage du Product Backlog. Cette assistance peut consister en des ateliers de conception anticipés, de priorisation ou d'estimation. Il faut compter environ 10% de la capacité à faire de l'équipe de développement pour ces activités.

Mêlée quotidienne ou « stand-up meeting »

Durée maximum : 15 minutes.

Cette réunion qui se fait debout (ça évite de s'éterniser) est très importante. Elle permet quotidiennement aux membres de l'équipe de se synchroniser, de remonter les obstacles rencontrés, de s'entraider, de vérifier l'avancement du sprint. Elle contribue également à faire naître l'esprit d'équipe. A condition bien entendu de ne pas transformer cette réunion de « synchronisation » en réunion de « reporting » vers le Scrum Master.

Photo d'une mêlée quotidienne

Chaque personne répond à 3 questions :

- Qu'ai-je fait hier qui a aidé l'équipe de développement à atteindre l'objectif Sprint ?

- Que vais-je faire aujourd'hui pour aider l'équipe de développement à atteindre l'objectif Sprint ?
- Est ce que je vois des obstacles susceptibles de m'empêcher ou d'empêcher l'équipe de développement d'atteindre l'objectif du Sprint ?

Le Scrum Master est ainsi immédiatement au courant des obstacles rencontrés, il doit impérativement les prioriser, les suivre et bien sûr s'efforcer de les lever au plus tôt afin de garder l'équipe pleinement concentrée et productive.

La mêlée quotidienne se déroule à lieu et heure fixes (devant le tableau des tâches physique de préférence) déterminés par l'équipe de développement. Au début le Scrum Master peut avoir à rappeler qu'il est l'heure de la mêlée et animer cette dernière en rappelant les 3 questions et évitant l'instruction des problèmes ou obstacles en séance afin de ne pas dépasser les 15 minutes imparties. L'objectif du Scrum Master consiste cependant à viser l'appropriation de la mêlée par l'équipe de développement.

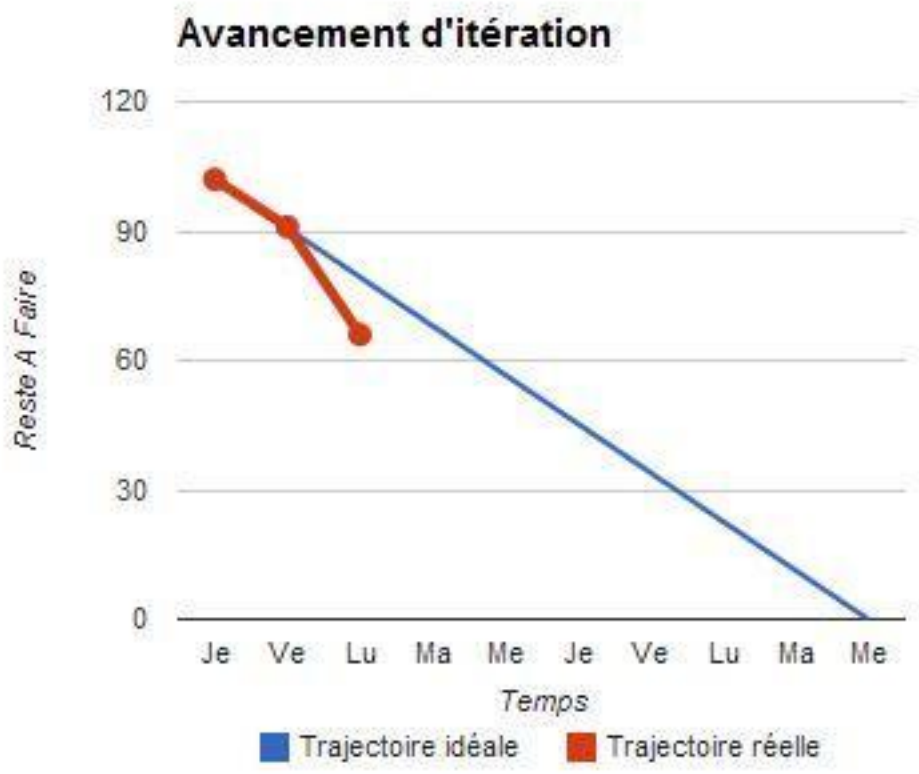
Graphique d'avancement (Burndown Chart)

Sprint Backlog

Exigence	Sous Tâche	Reste A Faire									
Exigence X - #102	IHM	8	8	6							
	Mise à jour documentation	2	2	0							
	Services métier	10	4	0							
	Tests automatisés Perf	8	8	8							
Exigence YY - #103	IHM	8	8	8							
	Mise à jour documentation	2	2	2							
	Services métier	10	10	5							
	Tests automatisés Perf	8	8	8							
Exigence XY - #33	IHM	8	8	8							
	Mise à jour documentation	2	2	2							
	Services métier	10	10	7							
	Tests automatisés Perf	8	8	8							
Exigence Y - #365	IHM	8	8	4							
	Services métiers	10	5	0							
	Jours d'itération	Je	Ve	Lu	Ma	Me	Je	Ve	Lu	Ma	Me
	Trajectoire idéale	102	91	79	68	57	45	34	23	11	0
	Trajectoire réelle	102	91	66							

Exemple de Sprint Backlog

Pour connaître votre avancement, vous allez avoir besoin de tracer le Burndown Chart du sprint en cours. Ce graphique est simple, il s'agit du tracé de la charge de travail restante (généralement en heures) en fonction du temps (en jours). Pour tracer ce graphique, il suffit de mettre à jour (lors de chaque mêlée quotidienne par exemple) le sprint backlog (voir illustration).



Exemple de Burndown Chart de Sprint

Vous pouvez également construire un indicateur d'avancement de Release à mettre à jour en fin de chaque sprint (voir outillage proposé dans le kit gratuit L'Agiliste).

Revue de Sprint

Photo d'une revue de sprint

Durée maximum : 1 heure par semaine de sprint (autrement dit : 2 heures pour des sprints de 2 semaines).

Fréquence : A la fin de chaque sprint.

L'objectif de la revue de sprint est d'inspecter l'incrément produit au cours du sprint écoulé, faire un point sur l'avancement de la Release et adapter au besoin le plan et le Product Backlog. L'équipe de développement présente à tout acteur projet intéressé (à minima le Product Owner idéalement accompagné d'utilisateurs finaux) les nouvelles fonctionnalités développées au cours du sprint. Le Product Owner donne un feedback à l'équipe de développement, il accepte ou refuse les fonctionnalités présentées.

L'équipe de développement calcule sa vélocité en additionnant les points d'estimation associées aux fonctionnalités acceptées. Une fonctionnalité partiellement terminée ne rapportera aucun point car une telle fonctionnalité n'est pas utilisable. La vélocité ainsi calculée va permettre de mettre à jour le graphique d'avancement de Release et de vérifier l'avancement de cette dernière. C'est l'occasion de vérifier que le nombre de sprints de la Release demeure adapté ou non.

La livraison à la fin de chaque sprint n'est pas obligatoire.

Rétrospective de sprint

Photo d'une rétrospective

Durée maximum : 45 minutes par semaine de sprint (autrement dit : 1 heure 30 pour des sprints de 2 semaines).

Fréquence : A la fin de chaque sprint.

Cette réunion est généralement animée par le « ScrumMaster » qui s'adresse à son équipe. Elle a pour but d'améliorer continuellement le processus de développement de l'équipe en mettant les idées de chacun à contribution. Il existe différentes techniques d'animation de rétrospectives. Voir le livre « Agile Retrospective: Making Good Teams Great » pour en savoir plus.

L'une d'elle consiste à identifier et pondérer les éléments positifs (éléments à cultiver ou source de motivation) du sprint écoulé, puis les éléments à améliorer, puis de définir un plan d'action d'amélioration (en commençant par améliorer les éléments dont la pondération est la plus forte). Si les membres de l'équipe sont à l'aise pour s'exprimer, un simple tour de table permet de remplir le paper-board d'éléments. Dans le cas contraire, on peut avoir recours aux post-it (chacun se voit remettre des post-it vierges et un marqueur et inscrit ses idées dessus pour ensuite les transmettre à l'animateur. Une idée par post-it.). Pour pondérer les éléments, il suffit d'allouer un certain nombre de points (5 par exemple) à chaque membre. Chaque membre peut ventiler ses points à sa guise (exemple : 4 points sur un sujet qu'il considère très important, 1 point sur un autre sujet également important à ses yeux mais quatre fois moins que le premier). Pendant la phase d'inventaire des éléments à améliorer, veillez à ne pas essayer de trouver des solutions avant la phase dédiée au plan d'action d'amélioration. Vous risqueriez de ne pas faire remonter à la surface la majorité des « problèmes ». Ce serait dommage car, j'ai pu constater que parfois, le simple fait d'évoquer un problème sans forcément avoir le temps de trouver une solution entraîne une amélioration au sprint suivant.

Commencer par les points positifs peut être vital lorsque le sprint a été éprouvant. L'équipe aura peut être besoin de se nourrir des éléments positifs avant de s'attaquer aux éléments à améliorer.

Tous les domaines peuvent être abordés en rétrospective : humain, organisationnel, pratiques, processus, outillage, qualité de vie au travail, conflits, interactions avec le métier. Le compte rendu de la dernière rétrospective, les graphiques d'avancement de sprint et release, les événements du sprint, les feedbacks de la revue de sprint sont autant d'éléments qui alimentent les conversations.

Projet Scrum et MOA

Il est important de prendre conscience que les méthodes Agile ne considèrent pas à juste titre de séparation MOA MOE contractuelle (typiquement française empruntée au secteur du BTP). Il est donc primordial de s'efforcer de décloisonner au maximum MOA et MOE, ces deux entités doivent collaborer étroitement tout au long du projet. Etant donné que l'on touche ici à des habitudes bien ancrées, ce chapitre explique quel rôle peut jouer la MOA dans un projet Agile.

Le rôle du consultant métier

Dans le cadre d'une approche Agile, les tâches d'un profil fonctionnel peuvent rester les même que dans le cadre d'une approche traditionnelle :

- Participation au reengineering des processus et à la conduite du changement.
- Modélisation des processus métier.
- Participation aux ateliers de conception.
- Rédaction des tests d'acceptation (en amont des développements).
- Vérification de la conformité des fonctionnalités (en aval des développements).

Le changement réside dans l'organisation du travail. L'approche séquentielle propice à l'effet tunnel est remplacée par un mode itératif, le travail est donc lissé dans le temps sur toute la durée du projet. De plus, un responsable métier devra jouer le rôle clairement défini de « product owner ».

Rôle du Product Owner

Le Product Owner (ou Directeur/Responsable Produit) est chargé de :

- Construire le Product Backlog.
- Ordonnancer ces dernières en fonction de leur importance métier.
- Répondre aux questions de l'équipe de développement.
- Valider/Rejeter une fonctionnalité « terminée ».
- Prendre des décisions importantes en temps voulu.

L'équipe de développement s'engage à faire une démonstration à la fin de chaque sprint le produit enrichi de nouvelles fonctionnalités. A la fin de chaque sprint, le Product Owner peut :

- Ajouter une exigence manquante.
- Retirer une exigence finalement inutile.
- Redéfinir la priorité des exigences.

Renfort du Product Owner

Le Scrum Master enseigne au besoin au Product Owner les règles du jeu de Scrum et la maîtrise des techniques associées. En particulier celles qui tournent autour de la gestion de Product Backlog, la planification de releases et l'utilisation des indicateurs de pilotage. Il facilitera également les interactions entre le Product Owner et l'équipe de développement. L'équipe de développement l'aide quant à elle à ordonnancer le Product Backlog en fonction des dépendances, risques et estimation qu'elle révèle. Enfin, une équipe de consultants métier peut l'aider sur :

- Les réponses à apporter aux questions de l'équipe de développement tout au long du projet.
- La prise de décisions.
- La rédaction des tests d'acceptation.
- L'ordonnancement du Product Backlog.
- La fourniture des feedbacks à la fin de chaque sprint.
- D'autres sujets : logistique, conduite du changement, déploiement, etc.

Par ailleurs, il est recommandé de solliciter quelques utilisateurs finaux représentatifs des utilisateurs visés par le produit afin de vérifier pas à pas (à minima en revue de chaque sprint) la bonne couverture de leur besoin.

Les pièges à éviter

Scrum est simple mais difficile

Nous l'avons vu, Scrum est un cadre de gestion de projet qui laisse à d'autres méthodes agile complémentaires, le soin d'apporter les pratiques de développement appropriées. C'est le cas de eXtreme Programming et software craftsmanship. Il est donc primordial de ne pas se contenter d'utiliser Scrum sans s'assurer que les pratiques de développement Agile sont ou seront maîtrisées in fine. Constatant de nombreuses dérives liées à cette erreur courante, les créateurs de Scrum alertent :

> *« Début 2009, davantage d'organisations utilisaient des processus agiles plutôt que des processus en cascade. Cependant, moins de 50% de celles utilisant Scrum développaient les itérations de façon incrémentale, ce qui est le cœur de Scrum. Un des plus grands défis de l'utilisation de Scrum a toujours été la courbe d'apprentissage des développeurs de l'équipe Scrum. »*

Gestion du changement

Selon la culture de votre organisation, Scrum peut être porteur de nombreux changements qui impliquent une véritable gestion du changement Agile.

Plusieurs équipes de développement Scrum

Idéalement, pour être efficace, une équipe de développement Scrum ne devrait pas dépasser 9 membres sans compter le Scrum Master et le Product Owner. Si votre projet nécessite plusieurs équipes de développement, évitez dans la mesure du possible de constituer des équipes par composant ou technologie (exemple : une équipe qui développe les IHM et une autre les services métier et l'accès à la base de données). Constituez des équipes pluridisciplinaires capables de développer intégralement une fonctionnalité (donc de coder toutes les couches). Comprenons nous bien, le principe n'est pas que chaque membre d'une équipe dispose de toutes le compétences mais plutôt que toutes les compétences requises soient couvertes par au moins un membre de l'équipe. De cette façon, chaque équipe peut avancer de façon autonome sans dépendre étroitement d'une autre, la coordination est plus simple. Chacune étant spécialisé sur un ou des domaines fonctionnels plutôt que technologiques. L'intégration du code de l'ensemble des équipes sera également plus facile.

Il y aurait davantage à dire sur l'utilisation de Scrum à grande échelle mais nous dépassons l'objectif de ce guide de démarrage.

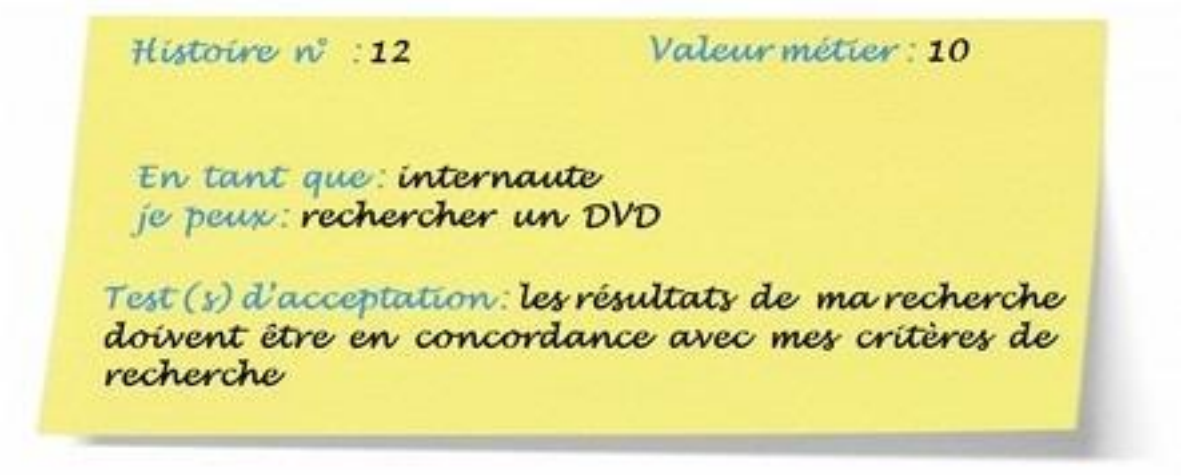
Annexes

Caractéristiques d'une User Story

Une fonctionnalité peut être rédigée selon le principe de la « User Story » défini par la méthode Agile eXtreme Programming (XP).

Une « User Story » doit être :

1. Courte : généralement une ou trois phrases environ.
2. Négociable : elle peut être discutée avec l'équipe chargée de la réalisation du produit, notamment lors de l'estimation.
3. Source de valeur : elle doit être porteuse d'une valeur pour le client ou l'utilisateur.
4. Indépendante des autres histoires d'utilisateur (dans la mesure du possible).
5. Estimable : elle peut être estimée par l'équipe de réalisation avec un risque d'erreur faible.
6. D'une taille appropriée : sa taille doit être relativement petite afin de faciliter son estimation. Elle doit pouvoir être conçue, développée et testée au sein d'une itération.



Exemple de User Story (simpliste pour l'exemple)

Quid des spécifications ?

La conception fonctionnelle voire technique d'une fonctionnalité se fait au fil de l'eau dans chaque itération. Ce n'est qu'au moment où on s'apprête à l'implémenter qu'on rentre dans les détails du besoin (pour rappel, le besoin lié à cette fonctionnalité à été abordé dans les grandes lignes lors de la phase amont d'estimation). Cette approche a le mérite de permettre au client d'ajuster ou approfondir son besoin sur les fonctionnalités implémentées plus tard. Elle lui permet aussi (ainsi qu'à l'équipe) de retarder certaines décisions sans risque pour le projet (c'est souvent plus sage quand on manque de visibilité).

La méthode Agile eXtreme Programming considère que les User Stories, les tests associés et le code source constituent les spécifications fonctionnelles du projet. D'autres méthodes vont plutôt préconiser la rédaction de cas d'utilisation avec un niveau de détail variable selon le contexte du projet, les exigences du client et la complexité de la fonctionnalité à spécifier.

Il faut bien comprendre que rédiger des spécifications ultra détaillées n'a de sens que dans une approche traditionnelle. Car dans ce cas de figure, le client ne verra son produit qu'à la fin des développements. Il doit donc avoir une vision précise de ce qu'il aura en fin de projet grâce aux spécifications détaillées. Or l'approche Agile apporte au client une visibilité sur le produit dès le début du projet ainsi qu'une possibilité d'ajustement du besoin. On privilégie donc le développement de l'application à une documentation exhaustive et pléthorique (d'ailleurs souvent indigeste pour le client qui tarde à valider et retarde ainsi son propre projet. Ou pire, valide sans vraiment lire et analyser).

Pour la partie technique on peut souvent se limiter à un dossier d'architecture, un modèle conceptuel de données (voir physique) émergeant, aux commentaires du code (Javadoc) et à une spécification détaillant les modules éventuellement complexes pour faciliter la maintenance du produit par la suite.
En appliquant ce principe, on assiste généralement à un gain de productivité par rapport à une approche traditionnelle, puisqu'on ne perd pas un temps considérable en rédaction, validation puis mises à jour de spécifications détaillées. Il est sans doute bon de préciser également que l'on ne néglige pas la conception du produit pour autant. La conception fait partie des tâches sous-jacentes à la réalisation d'une fonctionnalité.

Envie d'aller plus loin ?

Découvrez les formations et services proposés par Florent Lothon : http://www.agiliste.fr/formations/.

A propos de l'auteur

Florent Lothon

FLORENT LOTHON est expert en transformation agile, qu'il met en pratique depuis 2007. Il partage son expérience à travers son site http://agiliste.fr et ses formations certifiantes.

« Les méthodes agiles ont définitivement changé ma façon de mener et vivre un projet. J'ai créé le site L'Agiliste dans le but de partager cette expérience avec un maximum de monde. » - Florent Lothon

Du même auteur

Le livre "Devenir une Entreprise Agile".

Vous avez maintenant une bonne idée du potentiel que peut offrir l'agilité dans un contexte projet. Imaginez maintenant ce que l'agilité peut apporter à l'entreprise toute entière. Imaginez les atouts dont disposent les entreprises agiles dans le monde d'aujourd'hui. Un monde devenu complexe, incertain, avec de nouvelles générations de travailleurs aux attentes affirmées.

Découvrez la culture et les pratiques des entreprises agiles, au-delà de la gestion de projet agile. Intégrez l'agilité dans vos pratiques professionnelles pour transformer votre quotidien et votre entreprise.

Lien pour acheter ce livre