Chapitre 5
Construire et prioriser le Product Backlog

1. Pourquoi investir dans le Product Backlog ?

Que l'on soit en contexte Agile ou pas, il est assez évident que pour réussir un projet nous devons posséder une bonne vision du système ou produit que l'on va développer, par le biais d'une définition pertinente du besoin auquel il répond.

En Scrum donc, pas de projet réussi sans un Product Backlog bien construit et intelligemment priorisé. Pour ce faire, évidemment, le Product Owner joue un rôle fondamental, par sa connaissance du métier et/ou du marché, mais aussi par sa capacité à transcrire efficacement, découper et ordonner ce qui est attendu par les utilisateurs du logiciel.

Là où nous avions l'habitude en tant que « Client » d'écrire les spécifications fonctionnelles détaillées faisant office d'expression des besoins, nous allons devoir nous familiariser avec une nouvelle méthode. Cette familiarisation demande un temps d'apprentissage et se doit d'être partagée par l'ensemble des parties prenantes du projet, ce qui signifiera formation, assimilation et mise en pratique des nouvelles techniques.

Dans ce chapitre, nous allons donc parcourir les concepts et méthodes permettant de construire, ordonner, affiner et gérer au quotidien le Product Backlog.

2. La brique de base du Product Backlog : la User Story

Aussi étonnant que cela puisse paraître Scrum ne prescrit pas un format ou un contenu précis du Backlog ! Pour autant, un formalisme s'est imposé et on le retrouve quasiment systématiquement sur tous les projets Scrum : il s'agit de la notion de **User Story**, qui est empruntée à XP.

Une User Story est une description simple et compréhensible d'un élément de fonctionnalité à valeur « métier » du système. Elle est donc bien exprimée du point de vue de l'utilisateur. Et elle est guidée par la réponse à ces trois questions :

– Qui fait la demande ou qui bénéfice de la demande ? (rôle utilisateur)

– Quelle est la demande ? (le besoin)

– Quelle valeur métier découle de la réalisation de ce besoin ?

Ci-dessous quelques exemples (nous verrons plus loin comment bien rédiger les User Stories) :

« Je souhaite que la TVA soit automatiquement calculée sur les factures ».

« Je souhaite pouvoir supprimer les clients n'ayant pas passé de commandes depuis plus d'un an ».

En complément, on emploie aussi la notion d'**Epic Story**. On peut considérer une Epic comme une « **macro User Story** », c'est-à-dire qu'elle englobe dans sa définition un sous-ensemble de User Stories.

Par exemple, en relation avec les User Stories décrites précédemment, nous pourrions avoir les Epics suivantes :

« Je souhaite que mes factures soient établies automatiquement ».

« Je souhaite avoir une gestion de mes clients ».

C'est donc l'ensemble des Epic et User Stories que constitue le Product Backlog.

Il est en pratique utile de regrouper les Epic ou User Stories (en particulier pour la priorisation) : pour ce faire, on emploie communément les concepts de **Thèmes** ou **Activités**, qui sont des regroupements thématiques de Stories.

3. Comment rédiger les User Stories et Epics ?

3.1 Règle des 3C

Pour guider la rédaction des User Stories, un principe très simple à retenir a été proposé par Ron Jeffries. Il s'agit de la règle des 3C :

Carte	La Story est écrite sur une carte de taille assez réduite. Ces fiches peuvent être annotées (estimation, etc.).
Conversation	Les détails de la Story seront exprimés lors de conversations avec le Product Owner.
Confirmation	Des tests de validation sont décrits avec la Story (ils serviront à valider qu'elle a été réalisée correctement).

En pratique, on écrira donc la Story sur une petite carte de couleur colorée épinglée ou collée sur un tableau (principe hérité du Kanban, si vous vous souvenez bien), sous la forme suivante :

– On commence avec un titre.

– On ajoute une description synthétique de la « tâche utilisateur » et de ses objectifs.

– On peut y ajouter toutes les notes, dessins ou informations utiles. Exemple : chiffrage, indicateur de priorité ou valeur métier.

– On y ajoute idéalement les critères de validation (par exemple au dos si on n'a plus de place…).

Cela donnera, dans la forme la plus simple, quelque chose qui ressemble à ceci :

La carte est un concept de partage de l'information extrêmement synthétique et efficace, mais dès que des données additionnelles s'ajoutent, au fur et à mesure de l'analyse, attention à ne pas surcharger celle-ci. On voit assez vite venir la nécessité de passer par un outillage plus sophistiqué (et informatisé) pour gérer l'information : nous parlerons de cela plus loin…

3.2 Rédiger une bonne User Story : le principe INVEST

Lorsqu'on entre dans le processus de rédaction, on peut rapidement être perdu… Dans ce cas, un autre principe très utile vous guidera : il s'agit du principe INVEST.

Il indique qu'une bonne User Story doit être :

- **Indépendante** des autres histoires d'utilisateur (dans la mesure du possible).
- **Négociable** : elle doit pouvoir être discutée avec l'équipe chargée de la réalisation du produit, notamment lors de l'estimation.
- Source de **Valeur** : elle doit être porteuse d'une valeur pour le client ou l'utilisateur.
  - Une User Story ne décrit pas des finalités techniques !
- **Estimable** : elle peut être estimée par l'équipe de réalisation avec un risque d'erreur faible.
  - Elle doit, à cette fin, être rédigée de manière claire et compréhensible.
- D'une taille **Suffisamment petite** afin de faciliter son estimation et afin d'assurer qu'elle puisse être conçue, développée et testée au sein d'un Sprint.
  - Si la Story est trop grosse, c'est sans doute plutôt une Epic, qui devra être découpée plus finement préalablement à la réalisation.
- **Testable** : une User Story doit être accompagnée des critères de validation permettant sa validation.
  - Nous verrons dans le chapitre dédié aux tests comment formaliser cela.

3.3 Erreurs courantes

Pour bien rédiger nos Stories, il est aussi très instructif de savoir quelles erreurs éviter :

– Trop détailler la description et rentrer trop tôt dans un niveau de détail fin.

  N'oublions pas qu'on ne cherche pas à faire les spécifications détaillées complètes préalablement au développement comme dans les méthodes traditionnelles !

– Perdre la notion utilisateur dans la description ou utiliser des acteurs trop génériques : cela peut créer des ambiguïtés et imprécisions

  Exemple :

  – En tant que client abonné, je veux pouvoir consulter les tarifs des billets d'avion.
  – En tant que client standard, je veux pouvoir consulter les tarifs des billets d'avion.

  Plutôt que :

  – En tant qu'utilisateur, je veux pouvoir consulter les tarifs des billets d'avion.

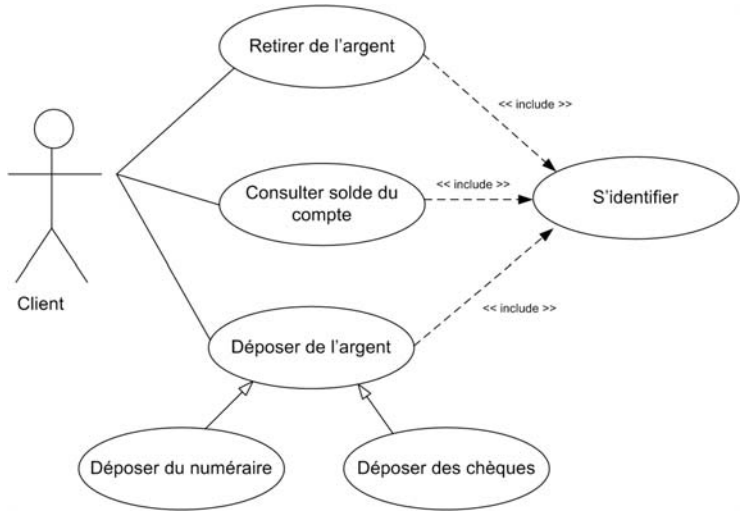
  En effet, on voudra vraisemblablement proposer des offres ou des tarifs différents aux abonnés par rapport aux autres personnes qui consultent les tarifs, ce qui n'apparaît pas dans la seconde description.

– Partir d'un cahier des charges rédigé en amont et le découper en User Stories en s'appuyant aveuglément sur sa structure textuelle. Cela peut être une approche commode pour une équipe débutante mais n'est pas une pratique recommandée (nous verrons plus loin une méthode pratique pour initialiser le Product Backlog).

– Vouloir avoir un niveau de détail constant. Ce point est très important, car il est au cœur de la démarche Agile. Le contenu des User Stories est amené à évoluer au fil du temps, au fur et à mesure de l'affinage de la compréhension du besoin et de l'analyse. Typiquement, une Story dont la réalisation est prévue dans plusieurs Sprints ne sera décrite que de manière très simple (titre, description synthétique), alors qu'une User Story dont la réalisation est proche doit être complétée par des tests de recette, des exemples, des règles de gestion, des maquettes d'écran, etc.

– Assimiler une User Story à un **Use Case**. Bien que la comparaison entre les deux notions soit souvent utile, ce sont deux approches aux caractéristiques sensiblement différentes.

Sans rentrer dans un niveau de détail trop fin, on peut dire qu'une modélisation à base de Use cases est une description de processus (par définition, un Use case représente une séquence d'actions qu'un système ou toute autre entité peut accomplir en interagissant avec les acteurs du système), qui s'appuie souvent sur des diagrammes UML, comme dans l'exemple ci-dessous :



3.4 La Story technique : solution ou aveu d'échec ?

Disons en préambule que ce thème est éminemment polémique dans la communauté Agile, car certains considèrent que le Product Backlog ne doit contenir que des éléments qui ont de la valeur d'un point de vue utilisateur ou métier alors que d'autres disent que finalement Scrum ne dit rien de précis à ce sujet, donc on peut inclure des sujets à vocation purement technique dans le Backlog.

Chapitre 1
Le rôle du Scrum Master

1. Ce que nous dit le guide Scrum

The Scrum Master is responsible for promoting and supporting Scrum as defined in the Scrum Guide.

Le Scrum Master a donc une double fonction, de promoteur et de supporter.

Cette définition du rôle du Scrum Master date de 2017. Dans la version précédente, il devait simplement s'assurer que Scrum était compris et mis en pratique.

Ce changement peut sembler anodin, mais il est très structurant car il définit plus précisément la manière dont le Scrum Master doit se comporter. Non pas comme un inquisiteur qui impose un dogme et condamne les déviants, mais plutôt comme un facilitateur.

1.1 Une fonction de promoteur

Un promoteur est un initiateur, un précurseur, celui qui donne une impulsion à quelque chose. C'est celui qui fait avant les autres. Il n'impose pas les processus Scrum, mais il en fait la promotion.

Ce choix n'est pas anodin. Il aurait été possible de définir le Scrum Master comme le gardien, le garant ou le contrôleur de Scrum. Dans ce cas, il aurait été légitime qu'il impose les processus empiriques, les piliers et les valeurs de Scrum, mais ce n'est pas le cas.

En tant que promoteur de Scrum, le Scrum Master doit pratiquer lui-même l'empirisme au quotidien. Il peut s'appuyer sur des théories ou des techniques pour convaincre, mais c'est au travers de l'empirisme qu'il parvient à réaliser au mieux sa mission, en démontrant de manière factuelle et empirique qu'il est bénéfique pour tous d'appliquer les processus de Scrum.

Il donne l'impulsion initiale et propose des mesures pour contrôler de manière empirique si les résultats sont conformes aux attentes.

Il s'appuie sur l'empirisme, mais il doit également incarner les valeurs de Scrum.

1.2 Une fonction de supporter

Le Scrum Master doit également soutenir Scrum. Lorsque tout va bien, cette tâche est aisée, mais c'est lorsque les choses se compliquent que le rôle de support prend tout son sens. Dans Scrum, la meilleure manière de résister aux tensions consiste à s'appuyer fermement sur les trois piliers de Scrum : Transparence, Inspection, Adaptation, et le Scrum Master doit être particulièrement vigilant vis-à-vis du premier des trois :

The Scrum Master's job is to work with the Scrum Team and the organization to increase the transparency of the artifacts. This work usually involves learning, convincing, and change. Transparency doesn't occur overnight, but is a path. (Transparence des artefacts, Guide Scrum page 17)

Le Scrum Master doit donc convaincre par l'exemple, et se placer au service de tout le monde pour promouvoir et supporter Scrum.

Comment remplir cette double fonction, là encore le guide Scrum donne quelques éléments précis :

Scrum Masters do this by helping everyone understand Scrum theory, practices, rules, and values.

Le point important est : « *helping everyone* ***understand*** ».

Il n'est pas demandé au Scrum Master d'enseigner, de délivrer son interprétation d'une réalité, tel un gourou, mais d'aider tout le monde à comprendre la théorie de Scrum, c'est-à-dire l'empirisme.

Le Scrum Master ne doit pas non plus imposer sa vision de l'empirisme, mais aider à comprendre cette théorie et la mettre en pratique. De la même manière, il doit aider tout le monde à comprendre pourquoi les trois piliers de Scrum, transparence, inspection et adaptation, soutiennent efficacement la mise en œuvre d'un contrôle empirique des processus.

Pour convaincre les autres, encore faut-il avoir soi-même parfaitement compris pourquoi les trois piliers de Scrum soutiennent l'empirisme et comment les cinq valeurs de Scrum créent de la confiance et font vivre ces trois piliers. C'est ce que nous découvrirons dans le chapitre Théorie et principes de Scrum, mais revenons à la description du rôle du Scrum Master.

The Scrum Master is a servant-leader for the Scrum Team.

1.3 Leader-serviteur

Aucune théorie du management n'avait jusqu'à présent évoqué ce style managérial de « leader-serviteur ». Nous abordons ici un sujet qui parfois divise : le Scrum Master est-il un manager ?

La réponse est clairement oui, mais pas du point de vue classique, c'est-à-dire hérité des concepts hiérarchiques basés sur le *command & control*. Le point de vue de Jurgen Appelo est plus élégant : « management is too important to leave to the managers, management is everyone's job ».

Sur ce point, le guide Scrum précise au chapitre concernant l'équipe de développement :

Development Teams are structured and empowered by the organization to organize and manage their own work.

L'équipe de développement n'est pas managée par le Scrum Master, mais par elle-même sous l'effet de la pression sociale. Nous aborderons ce point plus en détail dans le chapitre Le management participatif.

De la même manière, le Scrum Master a un rôle managérial. D'ailleurs, lors du passage de l'examen PSM I, vous pouvez être amené à répondre à cette question par oui ou non :

« Scrum Master is a "management" position? »

La bonne réponse est oui.

Le Scrum Master ne dirige pas les décisions de l'équipe, pas plus qu'il ne les influence. Il n'évalue pas les performances de l'équipe ni la capacité du Product Owner à organiser son Backlog de produit. En revanche, il aide l'équipe à mettre en place des processus efficaces d'auto-évaluation et il aide le Product Owner à gérer efficacement son Backlog de produit. Il n'a pas un rôle d'encadrement visant à faire respecter des processus et à sanctionner les éventuels divergents, mais un rôle de conseiller, de guide, ou de coach.

Au lieu d'imposer des changements, le Scrum Master aide tout le monde à comprendre pourquoi des changements sont nécessaires. Le quoi et le comment ne dépendent pas de lui. Il donne l'impulsion et convainc, notamment par l'exemple et en s'appuyant sur l'empirisme.

Le style de management le plus proche est celui décrit par Rensis Likert : le management participatif. Nous reviendrons plus en détail sur ce style de management dans le chapitre Le management participatif.

Le guide Scrum décrit un peu plus en détail le style managérial du Scrum Master dans les lignes qui suivent :

The Scrum Master helps those outside the Scrum Team understand which of their interactions with the Scrum Team are helpful and which aren't. The Scrum Master helps everyone change these interactions to maximize the value created by the Scrum Team.

Il aide tous ceux qui se trouvent en dehors de l'équipe Scrum à interagir plus efficacement avec elle en leur faisant comprendre comment rendre leurs interactions plus efficaces afin d'accroître la valeur créée par l'équipe Scrum.

1.4 Le Scrum Master Leader

Le Scrum Master n'est pas le chef de l'équipe, pas plus qu'il n'est le remplaçant du chef de projet. Il ne faut pas traduire Leader par chef ou dirigeant, mais plutôt inspirateur.

Scrum signifiant littéralement « mêlée », le Scrum Master a parfois été assimilé à un demi de mêlée. Si vous avez absolument besoin d'une comparaison, il est préférable de l'imaginer comme un coach sportif. Il ne code pas, ne conçoit pas, ne réalise pas, ne marque pas de points. Pourtant aucune équipe professionnelle ne joue sans entraîneurs.

Concrètement, le Scrum Master explique pourquoi il faut faire les choses, mais il n'impose pas ce qu'il faut faire et encore moins comment il faut le faire. En revanche, il peut suggérer des bonnes pratiques glanées dans ses expériences avec d'autres équipes ou bien auprès d'autres Scrum Masters.

Le style de management participatif proposé par Likert repose sur trois éléments :

– Le rapport de coopération entre les membres de l'organisation.
– La prise de décision et le contrôle par les groupes.
– La fixation à l'intérieur des groupes d'objectifs globaux et personnels ambitieux.

Ce n'est donc pas au Scrum Master de fixer des objectifs de Sprint à l'équipe, mais il doit faire comprendre à l'équipe pourquoi chaque membre a intérêt à participer à l'identification et la poursuite d'un but commun à chaque Sprint.

De la même manière, le Scrum Master incitera chaque membre de l'organisation à mettre en pratique les valeurs de Scrum lorsqu'il les incarne lui-même :

Focus sur les individus, en pratiquant une écoute réelle et sincère, en faisant preuve d'empathie. Focus aussi sur les engagements pris et les moyens réellement mis en œuvre. Enfin, focus sur les résultats escomptés et ceux réellement mesurés.

Ouverture, en cherchant à comprendre l'autre sur la base de faits au lieu de le juger sur la base d'émotions personnelles ou collectives.

Respect, en admettant qu'il ne détient pas la vérité. En acceptant momentanément que tout ne se passe pas comme cela le devrait en théorie. Le respect consiste également à pardonner au lieu de condamner, même et surtout lorsqu'on a prévenu un manager ou bien une équipe que leurs actions avaient peu de chances de produire les résultats escomptés.

Courage, surtout lorsqu'il faut affronter un échec personnel ou collectif, mais également lorsqu'il faut répéter le même conseil pour la centième fois avec la même authenticité et la même patience que si c'était la première fois.

Engagement, auprès de chaque membre de l'équipe et envers toute l'organisation. Faire preuve d'engagement nécessite de connaître ses limites et celles de l'équipe. Il faut s'engager au maximum, être ambitieux, sans être téméraire ni déraisonnable.

Le Scrum Master a bien un rôle de management, même s'il n'est pas un manager au sens hiérarchique du terme. Il n'impose pas, il inspire et convainc, tant par son comportement que par des éléments factuels. Pour ce faire, il adopte un style de communication assertive (voir chapitre Le Scrum Master, communiquant assertif).

1.5 Le Scrum Master Serviteur

Le guide Scrum décrit la mission du Scrum Master en tant que serviteur auprès de trois cibles :

– Le Scrum Master au service du Product Owner
– Le Scrum Master au service de l'équipe de développement
– Le Scrum Master au service de l'organisation

Nous reviendrons en détail sur ces trois missions aux chapitres correspondants, mais il apparaît clairement à la lecture des dix-sept fonctions décrites dans ces trois paragraphes que le Scrum Master ne peut pas tout faire en même temps, même s'il dédie 100 % de son activité à son rôle de Scrum Master. C'est-à-dire qu'il n'est pas membre de l'équipe de développement en plus d'être Scrum Master. En effet, bien qu'il soit fortement déconseillé que le Scrum Master endosse également la fonction de Product Owner, rien n'interdit qu'il soit également membre de l'équipe de développement. Hélas, dans ce cas, ce que l'on constate fréquemment, c'est que le Scrum Master s'efface progressivement pour se focaliser sur la réalisation des missions.

Ce phénomène est amplifié par les points suivants :

– La pression systémique : lors d'un Sprint particulièrement ambitieux impliquant un engagement accru de l'équipe de développement.
– La pression hiérarchique : lorsque le management, plutôt que d'arbitrer ou de prioriser en fonction d'une stratégie, souhaite essayer plusieurs solutions en même temps.
– La pression sociale : dès que l'équipe de développement est en difficulté.
– La présence d'un coach agile externe pouvant se substituer partiellement au Scrum Master.