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.

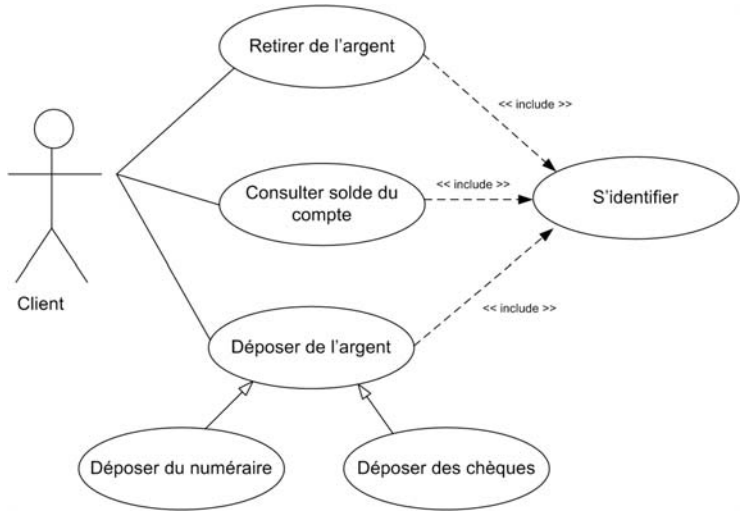
  En effet, on voudra vraisemblablement proposer des offres ou des tarifs différents aux abonnés par rapport aux autres personnes qui consultent les tarifs, ce qui n'apparaît pas dans la seconde description.

– 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.