



PRODUCT Academy

..... *Le guide*
des *Product Managers*
Product Owners et des
d'élite


VOLUME 1



THIGA et Xebia

Product Academy

Le guide des Product Managers
et des Product Owners d'élite



Ce livre a été écrit avec passion par les
Product Managers & Product Owners
de la société Thiga et les coachs Agiles
de la société Xebia.

TABLE DES MATIÈRES

À propos de nous	7
Introduction	8
Imaginer le produit	10
Règle n°1 : Travaillez votre vision produit	11
Règle n°2 : Ne résolvez pas des problèmes qui n'existent pas	17
Règle n°3 : Évaluez votre marché : l'étude d'opportunité frugale	22
Construire le produit	28
Règle n°4 : Construisez votre roadmap	29
Règle n°5 : Rédigez de belles user stories	37
Règle n°6 : Priorisez votre backlog.....	42
Règle n°7 : Préoccupez-vous de la dette technique.....	47
Règle n°8 : Plusieurs Product Owners pour une équipe de développement, comment prioriser ?.....	54
Règle n°9 : Adoptez la bonne posture	59
Faire évoluer le produit	64
Règle n°10 : Enrichir vs. Optimiser.....	65
Règle n°11 : Eat your own dog food.....	69
Règle n°12 : Gagnez de nouveaux utilisateurs.....	72
Règle n°13 : Gagner un nouvel utilisateur c'est bien, le faire revenir c'est mieux	79
Règle n°14 : Quand et pourquoi tuer son produit ?	85
Les auteurs	90
Remerciements	92

À PROPOS DE NOUS

Xebia et Thiga font toutes les deux parties de l'Alliance, un écosystème de sociétés complémentaires et expertes dans leur domaine. L'Alliance propose l'ensemble des compétences nécessaires à la mise en place d'une stratégie numérique cohérente et efficace, qu'elles soient technologiques, fonctionnelles ou méthodologiques.

Xebia

Xebia est un cabinet de conseil international spécialisé dans les technologies Big Data, Cloud et Web, les architectures Java et la mobilité dans des environnements agiles. Au travers de ses 4 valeurs (People First, Customer Intimacy, Sharing Knowledge, Quality without Compromise), Xebia se distingue par sa manière d'interagir avec ses clients et ses employés. Xebia est ainsi devenue une Great Place to Work en 2014.

Depuis sa création, il y a 10 ans, la mission de Xebia est d'être une autorité dans l'écosystème des nouvelles technologies.

THIGA

Thiga est un cabinet de conseil spécialisé en Product Management. La mission de Thiga est d'aider les entreprises à créer des produits numériques à succès. Pour atteindre cet objectif, les consultants de Thiga interviennent à deux niveaux :

- En coaching sur la phase de « Product Discovery » : en les accompagnant sur la création de leur Stratégie Produit/Vision Produit, Thiga aide ses clients à faire le pont entre la stratégie business de l'entreprise et les roadmaps produits.
- En réalisation sur la phase de « Product Delivery » : via l'intervention de Product Managers/Product Owners expérimentés, Thiga participe activement à la création des Produits de ses clients.

Thiga est aujourd'hui un des seuls cabinets de conseil français positionné exclusivement sur le Product Management.

Thiga, UX Republic, Xebia, Cellenza,
Xebia Labs, WeScale et Akamis



INTRODUCTION

En 2012, Gilles Mantel, Coach Agile de Xebia, publiait Scrum Master Academy, le guide des Scrum Masters d'élite. Ce livre, au ton volontairement décalé, avait pour ambition de donner aux Scrum Masters déjà expérimentés les clés pour être encore meilleurs dans leur mission.

Product Academy s'inscrit dans la lignée de son prédécesseur en s'adressant cette fois aux Product Managers et Product Owners, ces personnes, qui au quotidien, sont le bras armé de la stratégie numérique de l'entreprise.

Il est intéressant de noter que si le rôle de chef de produit existe depuis des années dans de nombreuses industries, il est relativement neuf dans le monde du numérique et il est souvent mal compris. Il s'agit pourtant d'un rôle crucial dans la réussite d'un produit et c'est pour cette raison que nous avons décidé de lui consacrer cet ouvrage.

Ce livre se veut concret et actionnable. Il est à destination des Product Managers et Product Owners qui souhaitent créer de nouveaux produits ou faire évoluer des produits existants, le tout dans un environnement agile.

Le livre s'articule en trois grandes parties qui reprennent les étapes du cycle de vie d'un produit numérique : imaginer, construire et faire vivre.

Au travers de quatorze règles, les lecteurs assidus pourront ainsi suivre la vie d'un produit de sa vision initiale jusqu'à son retrait du marché.

Chacune des règles de ce livre répond à une problématique concrète : Comment être sûr de faire le bon produit ? Comment construire une roadmap ? Comment prioriser ? Comment conserver mes utilisateurs ? Dois-je tuer mon produit ?

Au travers des pages de ce livre, nous vous livrons nos méthodes, nos outils et nos convictions. Nous espérons qu'ils vous aideront à créer les bons produits, ceux qui feront le succès de votre entreprise.


Bonne lecture !

Hugo Geissmann, président de Thiga

Imaginer le produit



Dans cette partie, nous allons vous expliquer comment challenger vos idées de produits numériques avant de vous lancer dans la phase de réalisation technique.



Règle n°1

TRAVAILLEZ VOTRE VISION PRODUIT



Construire un produit à succès est quelque chose de difficile. Il faut à la fois que le produit apporte une valeur réelle à ses utilisateurs et qu'il soit d'une qualité irréprochable. En d'autres termes, le leitmotiv d'un Product Manager pourrait être "do the right product and do it right".

L'introduction des pratiques agiles dans les départements IT a beaucoup contribué à l'amélioration de la qualité des produits. Cependant, l'agilité a encore du mal à franchir la porte des départements dits « métiers ». Ces derniers ont en effet tendance à voir l'agile comme un moyen de faire développer toutes leurs idées tout en s'autorisant à changer d'objectifs et d'avis en permanence.

Être agile, ne veut pas dire que toutes les idées doivent être réalisées sous prétexte qu'il est possible de le faire. Cette approche a pour conséquence de faire travailler les équipes de développement sur une succession de projets parfois sans cohérence business et sans capacité de se projeter... En un mot : sans vision produit !

Les entreprises ont considérablement progressé sur le "do it right" mais encore faut-il faire le bon produit : celui qui suscite l'enthousiasme des utilisateurs existants et qui permet la conquête de nouveaux utilisateurs. Pour se donner toutes les chances de réussir, il faut donc travailler sa « vision produit » !

Qu'est ce qu'une vision produit ?

La vision produit est la réponse opérationnelle à la stratégie business de l'entreprise. Elle fait le pont entre cette stratégie et la roadmap du produit (release plan, story map, backlog dans un monde agile). Il faut bien noter que le terme de vision produit n'est pas propre à l'agilité. Tous projets et produits se doivent d'avoir une vision claire, quelles que soient les méthodologies utilisées.

La vision produit est la réponse aux questions suivantes :

- Quel problème essayons-nous de résoudre ?
- Pour qui résolvons-nous ce problème ?
- Quelle solution apportons-nous à ce problème ?
- Comment allons-nous lancer ce produit ? (go-to-market)
- Quel est le modèle économique ?
- Comment allons-nous mesurer le succès de ce produit (revenu, métriques) ?

Avoir une vision produit claire permet entre autres :

- D'accorder les parties prenantes (marketing, finance, équipes techniques).
- De garantir la motivation des équipes.
- D'anticiper les choix technologiques en fonction de cette vision.

Dans une équipe agile, la vision produit est encore plus importante que pour une équipe classique : avant chaque sprint ou avant le développement de chaque fonctionnalité, les membres de l'équipe doivent se demander si leur travail répond bien au "Product Statement". Si ce n'est pas le cas, c'est que l'énergie n'est pas utilisée à bon escient.

Travailler sa vision produit : outils et méthodes

Il existe de nombreux outils permettant de matérialiser sa vision, nous en faisons une petite liste dans la partie “toolbox” de cette règle.

D'un point de vue méthodologique, nous vous conseillons de commencer par utiliser le lean canvas.

Le lean canvas, mis au point par Ash Maurya, se compose de deux grandes zones : la partie de gauche étant centrée sur le produit, la partie de droite sur le marché.



1. **Segment utilisateur** : qui sera la cible de notre produit ? Qui sont nos potentiels early adopters ?
2. **Problème** : quel est le top 3 des problèmes que nous cherchons à résoudre pour nous, early adopters supposés.

3. **Proposition unique de valeur** : la proposition de valeur est la raison qui doit pousser vos prospects à devenir des utilisateurs. En quoi résolvez-vous leurs problèmes ?
4. **Solution** : quelles sont les 3 grandes fonctionnalités qui vont répondre aux problèmes de nos early adopters ?
5. **Canal** : quels sont les canaux gratuits et payants que vous pouvez utiliser pour atteindre vos futurs utilisateurs ?
6. **Métriques** : quelles sont les métriques que vous allez mesurer pour valider ou invalider vos hypothèses ? (NPS, AARRR, chiffre d'affaires...)
7. **Coûts** : quels sont vos coûts fixes et variables ?
8. **Revenu** : quel est le modèle économique ? Comment allez-vous gagner de l'argent ?
9. **Avantages concurrentiels** : qu'est ce qui vous rend plus performant que vos concurrents pour traiter les problèmes identifiés ? (Connaissance technique, carnet d'adresse, marque).

Il est important de noter que le lean canvas n'est qu'un support à la réflexion. Il peut parfois être tentant de le remplir en une séance de travail de quatre heures puis de le prendre pour argent comptant tout au long de la création du produit. Même s'il a le mérite de donner une représentation claire de votre vision produit, l'essentiel du travail consiste cependant à itérer sur cette version initiale pour aboutir à un lean canvas dans lequel vous aurez fait le minimum de supposition. Chaque case doit donc faire l'objet d'un travail spécifique (cf règles 2 et 3).

Vision produit : combien de temps y consacrer ?

Investir du temps sur la création d'une vision produit n'est pas une garantie absolue de faire le bon produit. Cette vision évoluera nécessairement au cours de la vie du produit.

Des sociétés comme Twitter, PayPal, Pinterest ou encore Groupon ont toutes pivoté à plusieurs reprises avant d'avoir le positionnement que nous leur connaissons aujourd'hui.

Pinterest, par exemple, s'appelait initialement Tote et permettait à ses utilisateurs de parcourir les boutiques en ligne de leurs enseignes favorites et d'être prévenus de la disponibilité à la vente d'un article. Les fondateurs se sont rendus compte que leurs utilisateurs étaient plus intéressés par le fait de construire des collections à partir de leurs objets favoris puis de les partager à leurs amis. Ce constat les a donc poussés à faire évoluer leur plate-forme vers la version actuelle de Pinterest.

Même si elle peut évoluer dans le futur, bien cadrer la vision en amont reste le meilleur moyen de minimiser le risque d'un investissement humain et financier dans le développement d'un produit qui ne va pas trouver son marché.

La grande question est : combien de temps dois-je y consacrer ? Vous pourriez passer des mois à tenter de valider des hypothèses business ou faire des études in-situ. Évidemment, ce ne serait pas raisonnable. Pour bien calibrer cette phase, il faut avoir des ordres de grandeur en tête : d'expérience, combien de temps faut-il pour développer un produit du même type dans mon entreprise ? Chez mes concurrents ?

À titre d'exemple, nous réalisons généralement cette phase de vision pendant environ un mois pour des produits dont la durée initiale de développement (avant mise sur le marché) sera de huit mois à un an.

La toolbox

Nous avons déjà parlé du lean canvas, mais il existe de nombreux outils qui en sont des variantes :

- Product Vision board
<http://www.romanpichler.com/tools/vision-board/>
- Business Model Generation
businessmodelgeneration.com
- Outils en ligne
<https://leanstack.com/>
- Validation board
<https://www.leanstartupmachine.com/validationboard/>

Règle n°2 :

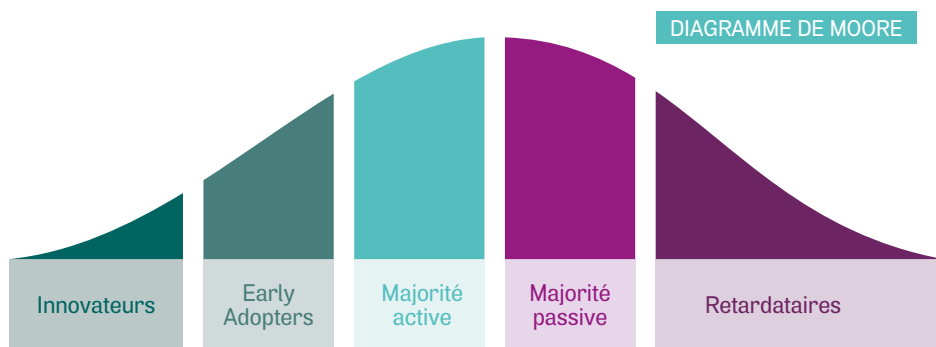
NE RÉSOLVEZ PAS DES PROBLÈMES QUI N'EXISTENT PAS



De nombreux produits ne trouvant pas leur marché finissent par disparaître. Les causes de l'échec sont évidemment variées mais une raison revient souvent : le produit ne résout aucun problème !

Pour éviter cet écueil et mettre toutes les chances de votre côté, nous vous conseillons de remettre en cause tout ce qui peut vous sembler évident de prime abord. Pour cela, explicitez les principaux problèmes que vous souhaitez résoudre et considérez les comme de simples hypothèses.

Avant de challenger vos hypothèses, ayez en tête que la première version de votre produit (aussi appelée MVP - Minimum Viable Product) devra répondre aux principaux problèmes d'une cible de niche (qui seront les probables early adopters du produit). Une fois le succès avéré pour ce produit de niche, il s'agira d'adapter votre produit pour aller conquérir une cible plus large (correspondant à la "early majority" du diagramme de Geoffroy A. Moore¹). Même Facebook, qui adresse aujourd'hui le grand public, a commencé par cibler les étudiants de Harvard, avant de s'élargir à d'autres universités américaines puis aux lycées, etc. en adaptant son produit à chaque occasion. C'est l'état d'esprit "Think big. Start small." (par opposition à la philosophie "Get Big Fast" du début des années 2000 consistant à lancer rapidement un produit industrialisé sans en avoir validé, au préalable, correctement l'appétence qu'en auraient les clients).



Pour réaliser un "crash test" de vos hypothèses en un temps restreint, vous pouvez commencer par recueillir l'avis d'experts du marché auquel vous vous attaquez et de Product Managers adressant la même cible que vous. Vous en tirerez des enseignements précieux qui vous amèneront à reformuler vos problèmes. Une fois les hypothèses affinées (ou complètement revues), il est maintenant temps d'aller parler à ceux qui seront peut-être vos futurs clients.

Dans certaines organisations, confronter ses idées sur le terrain avant de se pencher sur la solution est complexe : les commerciaux bloquent l'accès aux clients, le département communication ne veut pas abîmer la marque avec des discussions « non maîtrisées », les investisseurs souhaitent sortir le produit rapidement pour maximiser leur ROI, les développeurs veulent commencer à coder, etc. Le Product Manager devra donc faire preuve de doigté pour

1. http://www.blue-blog.fr/wp-content/uploads/2011/04/diagramme_moore11.jpg

contourner ces obstacles et être force de proposition auprès de sa hiérarchie pour rendre les processus de création de produits plus “user centric”.

Toutes les techniques décrites dans la suite de cette règle vont permettre au Product Manager de remplir les cases Problèmes, Early adopters, Promesse client et Solution du lean canvas.

Les early adopters, qui sont-ils ?

Persona

Un pré-requis pour tester ses hypothèses est de constituer un panel d'early adopters avec lesquels vous pourrez interagir tout au long du processus de création de votre produit et après le lancement. Si la promesse du produit est suffisamment forte, les early adopters vous aideront à définir la solution adéquate et évangéliser le marché.

Pour bien cerner vos early adopters, nous vous recommandons de les personifier à l'aide de personas (les attributs caractérisant vos personas varient grandement selon la nature des produits : données démographiques, catégorie socio-professionnelle, style de vie, appétence aux nouvelles technologies, etc.). De manière générale, référez-vous à ces personas, qui seront amenés à évoluer au fil des retours terrain, à chaque itération sur la vision de votre produit.

Constitution du panel

Pour démarrer la constitution du panel, le moyen le plus efficace consiste à identifier les early adopters potentiels dans votre entourage. Par recommandation, vous atteindrez facilement les N+2 voire N+3 de votre réseau. Pour compléter le panel, allez chercher les early adopters là où ils se réunissent : sur internet (réseaux sociaux, forums spécialisés, etc.) ou physiquement (les salons professionnels dans les cas des produits B2B, etc.). Enfin, un autre moyen extrêmement efficace – mais payant – est le recours à un panéliste (société spécialisée dans le recrutement de prospects) de type Ipsos, TNS Sofres, etc. Comptez quelques centaines d'euros par contact si votre cible est difficile à atteindre.

Une fois le panel constitué, il faut chercher à valider ou invalider les hypothèses par une série d'interviews orientées problèmes. Nous vous conseillons fortement de ne pas externaliser cette étape. Nous avons, en effet, constaté que les Product Managers collectant et analysant eux-mêmes les retours clients identifient des solutions bien plus pertinentes.

Les early adopters : leurs problèmes, vos solutions

Interviews orientées problèmes

Pour démarrer, privilégiez les interviews 1:1. Si vous êtes néophyte en conduite d'entretien, vous pouvez construire et suivre un script d'interview adapté à votre produit (comme celui proposé par Ash Maurya dans Running Lean²). Le script vous permet de confronter votre vision des problèmes avec celle des interviewés et d'identifier les solutions existantes qu'ils utilisent (le cas échéant). Surtout ne cherchez pas à vendre votre produit, vous êtes là pour apprendre !

Pour aller plus loin, de nombreux outils issus du monde du design, tels que les observations in situ des solutions existantes, vous aideront à identifier les comportements et usages réels de vos utilisateurs.

Passez chaque problème en revue et cherchez à évaluer son importance aux yeux du prospect. Attention, ne vous satisfaites pas du déclaratif, tentez de déceler la vérité. Par exemple, si un interviewé vous dit qu'il rencontre régulièrement un grave problème mais qu'il ne cherche pas activement une solution, c'est qu'il vous ment, inconsciemment ou pas, pour vous faire plaisir. De manière générale, vous savez que vous vous attaquez à de vrais problèmes quand un nombre significatif d'interviewés (environ 10-20) cherchent activement une solution ou, encore mieux, ont bricolé une solution et payent pour celle-ci.

Imaginez les solutions

Une fois que vous êtes sûr que vous adressez des problèmes qui valent la peine d'être résolus, c'est à vous de définir des pistes de solution. En effet, les personnes sont globalement douées pour décrire leurs problèmes mais beaucoup moins pour imaginer les solutions correspondantes.

2. https://s3.amazonaws.com/StartitUp/problem_interview.jpg

Selon nous, le meilleur moyen d'identifier des solutions pertinentes est d'appliquer les grands principes du Design Thinking :

- Réunissez une équipe pluridisciplinaire (incluant notamment un technologue, en charge de proposer des technologies permettant de transcender l'usage et l'expérience des solutions imaginées).
- Divergez intellectuellement en vous basant sur toute la matière inspirante à votre disposition (comprenant, bien entendu, les comptes-rendus des entretiens individuels).
- Convergez vers quelques pistes de solutions. Selon le contexte, les critères de la grille de convergence peuvent varier sensiblement : appétence client, cohérence avec la stratégie et l'univers de marque, faisabilité technique (estimée grossièrement en taille de T-shirt par exemple), monétisation espérée, capacité à breveter, barrières à l'entrée, etc.

Prototypiez...

Les pistes de solution que vous avez déterminées doivent par la suite être prototypées (idéalement à moindre coût c'est-à-dire avec un minimum de code). Les possibilités qui s'offrent à vous pour prototyper sont nombreuses : pièce de théâtre, storyboard (bande dessinée), vidéo décrivant un cas d'usage majeur de votre produit, maquettes (réalisées avec des logiciels de type Axure, Balsamiq ou PowerPoint). Dans tous les cas, nous vous recommandons de ne pas externaliser intégralement sa réalisation car il vous sera difficile de le modifier rapidement et au fur et à mesure de vos tests.

Et validez vos hypothèses

Armé de vos prototypes, recontactez les personnes que vous avez interviewées et testez vos solutions. L'objectif est de voir si les solutions imaginées répondent bien aux problèmes identifiés et, idéalement, de transformer ces interviewés en clients. Lors des interviews, demandez ce qui semble indispensable, superflu ou manquant. Pour vous assurer que la personne en face de vous est vraiment attirée par la solution (et ne vous dit pas que vos prototypes sont fantastiques uniquement par politesse), cherchez à obtenir un engagement concret. La précommande de votre produit (comme sur les sites de crowdfunding type Kickstarter) est évidemment la meilleure forme d'engagement mais il en existe beaucoup d'autres (accès aux données personnelles, au carnet de contacts...). En somme, tout ce qui a de la valeur pour vous et surtout pour vos prospects.

Règle n°3 :

ÉVALUEZ VOTRE MARCHÉ : L'ÉTUDE D'OPPORTUNITÉ FRUGALE



Les parties Problème et Solution de votre vision produit ont été définies et validées qualitativement. Vous avez également une première idée de votre modèle économique, définie dans votre lean canvas : principaux postes de coûts, modèle de revenus, prix de vente envisagé, canaux de distribution ou d'acquisition clients...

La tentation de se lancer directement dans la phase de développement est forte. Néanmoins, avoir un bon produit ou une fonctionnalité très attractive ne garantit pas le succès. À ce stade, votre modèle économique reste fondé sur de simples hypothèses, qu'il conviendrait de tester avant d'investir lourdement dans la création du produit lui-même.

Comment tester les hypothèses prises pour construire votre modèle économique et s'assurer que celui-ci résistera à l'épreuve des faits ?

Les grandes entreprises financent une « étude d'opportunité » complète, qui s'appuie sur une analyse des ressources disponibles en interne, une étude de marché et des projections de revenus pour constituer un dossier d'investissement permettant aux décideurs de prendre une décision de GO / NO GO.

Cependant, réaliser ce type d'étude coûte cher et peut conduire à repousser de plusieurs mois le début des développements ; quant à la réutilisation d'études existantes disponibles sur étagère, elle risque de donner des résultats imprécis voire incohérents, aucune étude ne couvrant à elle seule l'ensemble du périmètre ciblé par le produit.

Que l'on soit Product Manager dans un grand groupe ou dans une startup, il existe des outils et astuces pour valider les grandes hypothèses de son produit (besoin client, solution apportée, modèle économique, marché cible, etc.) de manière simple et agile.

Qualifiez l'opportunité en interne

Avant même de s'intéresser à la taille du marché cible ou au modèle économique de votre produit, il nous semble important de se poser les bonnes questions sur la capacité de votre entreprise à créer le produit et assurer son succès :

- Êtes-vous réellement capable de réaliser ce produit ? Il s'agit ici d'évaluer les compétences dont vous disposez, les briques technologiques sur lesquelles vous pouvez compter ou encore votre capacité à adresser les canaux clients que vous avez identifiés.
- Pourquoi êtes-vous le mieux placé pour mener à bien ce projet ? L'idée consiste à identifier vos avantages compétitifs (Unfair Advantage dans le framework du lean canvas) : base clients existante, contrat d'exclusivité avec un partenaire, capacité à sécuriser un canal d'acquisition de clients, compétences particulières valorisables, etc. Aucun produit ne peut cumuler l'ensemble de ces atouts, essayez malgré tout d'en acquérir au moins un pour assurer le succès de votre produit (à défaut, être le premier sur le marché reste un bon moyen de sécuriser un tel avantage).
- Le produit est-il cohérent avec ma stratégie d'entreprise ou de marque ?
- Est-ce le bon moment pour lancer un tel produit ?

Évaluez la taille du marché cible

En supposant que votre produit réponde à un besoin avéré, il reste à estimer le nombre de clients potentiels partageant ce besoin. En effet, il est possible que votre vision produit restreigne son usage à une catégorie précise de la population. Estimer le marché adressable de votre produit, même de manière imprécise, est précieux ; cela permettra de vérifier que votre modèle économique tient la route et guidera votre approche de l'acquisition client (on n'adresse pas de la même façon un marché de niche et un marché de masse).

Le market sizing est une méthode simple permettant d'évaluer le nombre maximal de clients potentiels pour son produit ou service, voire, en poussant l'exercice plus loin, une estimation du potentiel revenu du produit (selon l'équation : **revenu = marché adressable * part de marché * nombre d'achat par client * prix de vente**).

Il existe deux principales approches pour un market sizing :

- “Top-down” : en partant de la population totale, il suffit de la restreindre en appliquant différents critères socio-démographiques ou d'usage, jusqu'à atteindre le segment de marché cible. Par exemple, pour une application smartphone permettant de partager et télécharger des corrigés de baccalauréat payants, l'entonnoir pourrait se présenter de la façon suivante : **« jeunes de 17-18 ans > scolarisés > utilisateurs de smartphone > disposant d'une carte bleue »**. Dans cet exemple, nous voyons tout de suite que le marché adressable est probablement très faible. Pour être complet, nous pouvons considérer une seconde population, celle des parents achetant des exercices pour leurs enfants bacheliers.
- “Bottom-up” : cette approche est pertinente dans le cas d'un marché déjà adressé par des concurrents bien établis car elle nécessite de connaître ou de pouvoir estimer le nombre d'utilisateurs de ces concurrents. En additionnant leurs bases clients respectives, on aboutit ainsi à une vision du marché adressable.

En multipliant ce marché adressable par un pourcentage représentant le nombre d'utilisateurs que vous estimez pouvoir séduire chaque année (part de marché cible), vous obtenez votre nombre de clients potentiels.

Si celui-ci s'avère faible, ce n'est pas forcément signe qu'il faut abandonner le produit ; il faut simplement s'assurer que votre modèle économique est pertinent pour un tel marché de niche. Il n'y a pas de règles strictes en la matière mais on peut généralement considérer qu'un modèle économique fondé sur les micro-transactions ou basé sur la publicité aurait moins de chances de fonctionner en ciblant un marché restreint.

À l'inverse, se lancer sur un marché large, mature et déjà adressé par de nombreux acteurs n'est pas forcément rédhibitoire ; mais pour avoir une chance de réussir, il faut que votre produit soit différenciant par rapport à la concurrence. On peut citer en exemple Dropbox, lancé en 2008 sur un marché du stockage en ligne en pleine croissance mais déjà saturé d'acteurs ; ce qui ne les a pas empêché de s'imposer comme l'un des leaders de ce marché, grâce à un produit fiable et simple d'utilisation.

Testez l'appétence du marché

Pour vérifier si le produit ou la fonctionnalité envisagé(e) correspondent à une réelle demande des clients, certaines entreprises s'appuient sur des systèmes de précommande ou de crowdfunding. L'idée consiste autant à récolter de l'argent en amont pour financer la première version du produit (Minimum Viable Product) qu'à évaluer la demande potentielle. Par exemple, le projet de montre connectée de la startup Pebble a connu un gros succès sur Kickstarter (plus de 10M\$ collectés), leur permettant d'une part de valider l'existence d'un besoin marché non adressé et, d'autre part, d'obtenir un volume de précommandes suffisant pour optimiser les coûts de production du premier lot de produits. Plus récemment, en France, la newsletter d'actualité Brief.Me a lancé une campagne de financement participatif sur Ulule en assumant explicitement un double objectif : d'un côté tester l'intérêt pour le produit, recueillir des réactions et, de l'autre, construire une première base d'abonnés.

Une autre possibilité pour tester l'appétence du marché à moindre coût est la technique du smoke test. L'idée consiste à faire « comme si » la fonctionnalité existait déjà et d'évaluer les réactions des utilisateurs, grâce à des métriques définies en amont. Le smoke test peut prendre la forme d'un faux bouton, d'une publicité Google Adwords décrivant de manière synthétique votre proposition de valeur ou d'une fausse page d'accueil pour un produit ou fonctionnalité inexistant. Dans ce cas, on peut également parler de "fake door" : la fonctionnalité n'existe pas vraiment et la demande de l'utilisateur ne peut pas aboutir.

En évaluant le pourcentage des utilisateurs qui cliquent sur le bouton, visitent la landing page ou laissent un feedback, on peut avoir une estimation du succès futur de cette fonctionnalité. Afin que l'expérimentation soit fiable d'un point de vue statistique, il faut bénéficier d'une base utilisateurs suffisamment conséquente ; auquel cas, cette métrique vous permettra d'affiner l'estimation de part de marché utilisée durant la phase de market sizing.

Attention à ne pas rompre la confiance de l'utilisateur ou abîmer l'image de votre produit dans le cadre d'un smoke test : le risque de déception est grand si l'utilisateur clique sur un bouton pour s'apercevoir que la fonctionnalité n'est pas disponible. Mieux vaut être clair avec l'utilisateur en lui expliquant qu'il s'agit d'une fonctionnalité prévue dans un futur proche et l'inviter à donner son avis, plutôt que de feindre une erreur technique.

Certains types de smoke test nécessitent de construire une partie du produit tout en réalisant de manière non automatisée (à la main), la partie la plus complexe de la fonctionnalité. Zappos, un site américain de vente de chaussures en ligne, est souvent cité en exemple pour la façon dont le fondateur a utilisé un smoke test pour convaincre ses premiers investisseurs de la viabilité de son modèle : en réalisant un faux site d'e-commerce comportant une sélection réduite de produits, et en traitant chaque commande lui même en allant acheter les chaussures dans des magasins spécialisés avant de les expédier à ses clients. Cette approche est plus concrète que la précédente (on évalue l'appétence de l'utilisateur sur la base d'une vraie fonctionnalité tangible plutôt que d'une promesse) mais ne fonctionne qu'avec des volumes très restreints et n'a donc pas une grande valeur d'un point de vue statistique.

Par ailleurs, la technique du smoke test ne doit pas remplacer l'idéation produit ; il s'agit d'un outil de convergence, pas de divergence : tester tout et n'importe quoi se révèle à terme contre-productif.

Construire *le produit*

Maintenant que vous avez validé les grandes lignes de votre vision, nous allons vous guider pas à pas dans la construction de votre produit.

Règle n°4 :

CONSTRUISEZ VOTRE ROADMAP



Qu'est ce qu'une roadmap ?

La roadmap est l'itinéraire simplifié entre là où se trouve votre produit aujourd'hui (qu'il soit nouveau ou que vous envisagiez une n-ième itération) et la destination définie dans votre vision. Cet itinéraire est de haut niveau : il propose une suite d'étapes intermédiaires et les dates prévisionnelles correspondant au franchissement de ces étapes.

La roadmap comporte principalement des objectifs chiffrés associés aux moyens à mettre en oeuvre pour les atteindre, ainsi que les risques identifiés. Ne la confondez pas avec une liste sans fin de fonctionnalités prévues pour votre produit.

L'outil "Go Product Roadmap" de Roman Pichler constitue un bon support pour représenter une roadmap produit. Celui-ci présente pour chaque release :

- la date de livraison,
- l'objectif ou le thème de la livraison,
- les fonctionnalités principales incluses,
- les métriques à suivre.

GO PRODUCT ROADMAP					
Date	12/04/15	Juin 2015	Q3 2015	Q4 2015	S1 2016
Nom release					
Objectif					
Fonctionnalités					
Métriques					

Les risques et dépendances ne sont pas identifiés dans cet outil, mais il s'agit d'une bonne pratique ; n'hésitez pas à les y insérer. De même, votre ligne « métriques » devrait idéalement comprendre un certain nombre d'objectifs chiffrés.

Comment construire sa roadmap ?

Pour construire votre roadmap vous pouvez vous appuyer sur plusieurs outils qui ont déjà fait leurs preuves : la story map et l'impact map.

- Le story mapping est un exercice inventé par Jeff Patton, qui dans sa version basique, permet d'identifier et de structurer les fonctionnalités suivant deux axes : un axe (horizontal) découpant le parcours utilisateur en étapes chronologiques et un axe (vertical) détaillant ces étapes en fonctionnalités.
- L'impact mapping est une méthode visuelle de planning stratégique proposée par Gojko Adzic. Cette approche permet de réconcilier une vision globale orientée objectifs (apportée par les méthodes de projet classiques) et une vision orientée fonctionnalités (proposée entre autres par les méthodes agiles).

L'utilisation conjointe de ces deux outils vous permettra de construire une roadmap pertinente.

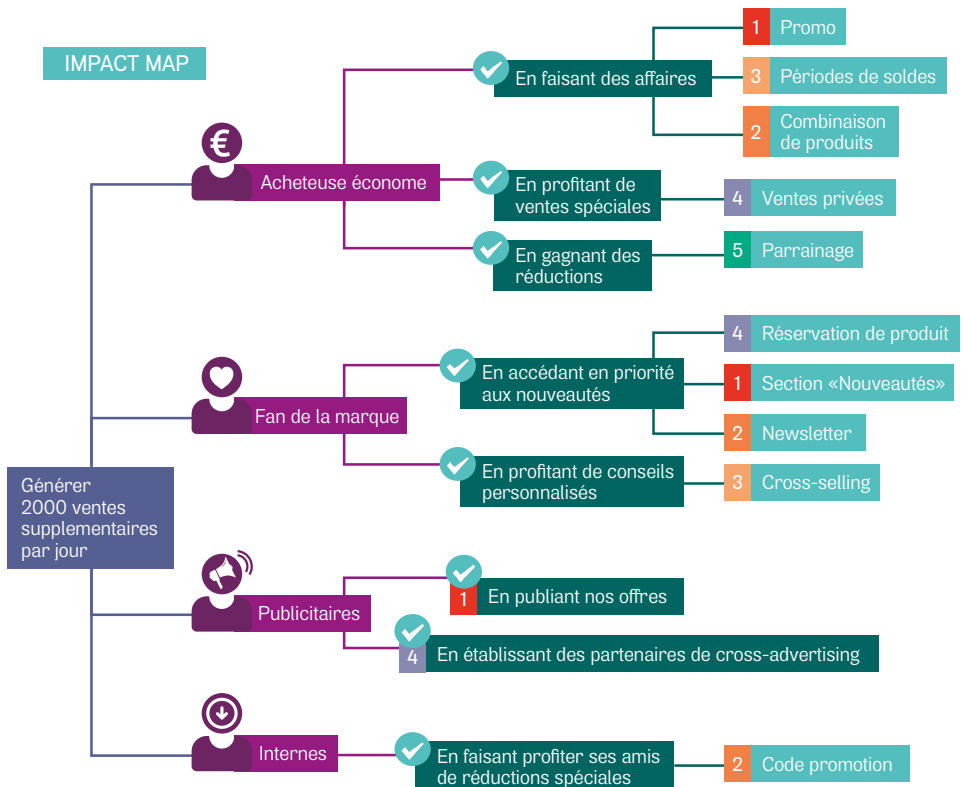
Faites un impact mapping

Comme expliqué précédemment, l'impact map vous permettra de faire découler les fonctionnalités des objectifs, et non le contraire. La création d'une impact map déroule les sujets de la façon suivante :

- **Les objectifs** : Pourquoi souhaitez-vous créer ou faire évoluer votre produit ? Dans votre vision initiale, matérialisée par un lean canvas, vous avez normalement défini des KPIs qui vous permettront de suivre vos objectifs business. Le point de départ de l'impact map est d'attribuer une valeur cible à ces KPIs. Par exemple : «atteindre 100 000 utilisateurs actifs mensuels» ou «obtenir la fidélisation de 30 000 utilisateurs».
- **Les acteurs** : De qui avez-vous besoin pour atteindre vos objectifs ? Quelle est votre audience ? L'impact map met en évidence deux types d'acteurs : ceux qui peuvent influencer sur votre objectif de façon indirecte (des publicitaires ou des fournisseurs par exemple) et ceux qui utilisent votre produit (vos utilisateurs finaux).

- **Les impacts :** Comment souhaitez-vous modifier le comportement des acteurs afin d'atteindre votre objectif ? C'est une étape complexe, car il ne s'agit pas seulement de décrire les usages de votre produit, il faut aussi établir les changements que vous souhaitez y opérer afin de vous rapprocher de votre but.
- **Les fonctionnalités :** Quelles fonctionnalités ou actions devez-vous mettre en place pour obtenir les comportements identifiés ? La granularité est ici celle du parcours utilisateur.

Voici un exemple d'impact map d'un site de e-commerce déjà opérationnel mais souhaitant développer de nouvelles fonctionnalités permettant d'accroître les ventes.



Dans cet exemple, l'acheteuse économe contribuera à notre objectif consistant à générer deux cents ventes supplémentaires par jour en profitant de ventes spéciales attractives, qu'il conviendra de mettre en place. Cette approche a pour principal intérêt de visualiser sur le même schéma les objectifs stratégiques, les différents types d'acteurs du produit et les fonctionnalités à développer pour que chacun de ces acteurs contribue aux objectifs. Une fois cette cartographie réalisée, vous aurez une bonne base de réflexion pour entamer le story mapping.

À noter que le fait de créer une impact map n'est pas une étape systématique, mais un moyen intéressant d'aborder la création de la roadmap sous un angle orienté « objectifs ».

Construisez votre story map

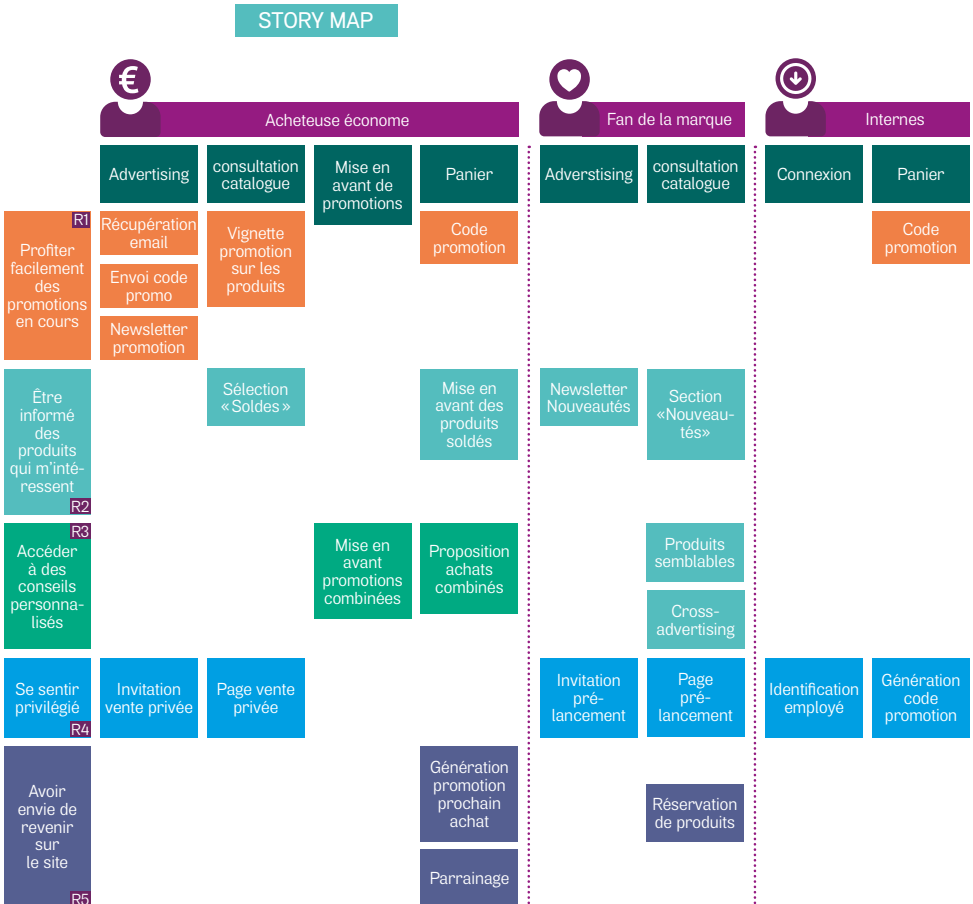
Lors de l'atelier de story mapping, il est important que toutes les parties prenantes soient présentes : marketing, métier, design, technique, etc. Cela vous permettra d'une part de bien prendre en compte les attentes et les contraintes de chacun, et d'autre part d'obtenir un consensus et un engagement sur votre roadmap. Nous vous recommandons de ne pas faire durer cet atelier plus d'une demi-journée.

Commencez par ré-introduire la vision, les objectifs et les personas, puis initiez la story map en inscrivant ces derniers sur le mur (sous forme de post-it par exemple). Reprenons l'exemple de la vente en ligne. Nous avons pour objectif de booster les ventes journalières, et avons identifié trois personas : l'acheteuse économe, la fan de la marque et l'employé. Chacun de ces acteurs utilise le produit d'une façon différente. L'enjeu consiste à dégager les étapes clés de leurs parcours utilisateurs et de réfléchir à comment les optimiser pour influencer positivement l'usage du produit.

S'ensuit une phase de discussion sur les fonctionnalités à mettre en place. À ce moment de l'exercice, il ne s'agit pas de restreindre les idées, mais d'envisager toutes les possibilités.

Dans un deuxième temps, vous pouvez regrouper les fonctionnalités en MMFs (Minimum Marketable Features) puis ordonner ces dernières pour définir des releases. Ce sont des itérations du produit, apportant chacune une nouvelle valeur supplémentaire à l'utilisateur et ayant une cohérence fonctionnelle.

Pour notre site de vente en ligne, nous pourrions proposer la story map suivante :



Nous arrivons à la dernière étape, la création de la roadmap proprement dite à partir de la story map. En pratique, vous allez extraire cette roadmap de la story map, chaque MMF correspondant à une ou plusieurs releases. Les premières MMF seront déjà très claires et précises, tandis que les suivantes resteront à l'état d'idée. Voici ce que cela donnerait dans le cadre de notre exemple :

GO PRODUCT ROADMAP

Date	12/04/15	Jun 2015	Q3 2015	Q4 2015	S1 2016
Nom release	Release 1
Objectif	Profiter des promotions en cours	Être informé des produits qui m'intéressent	Accéder à des conseils personnalisés	Première sortie privilégiée	Fidéliser
Fonctionnalités	<ul style="list-style-type: none"> ▶ Mise en avant du produit ▶ Code promo 	<ul style="list-style-type: none"> ▶ Soldes ▶ Nouveautés 	<ul style="list-style-type: none"> ▶ Promo combinées ▶ Cross-selling 	<ul style="list-style-type: none"> ▶ Ventes privées ▶ Pré-lancement ▶ Code employé 	<ul style="list-style-type: none"> ▶ Fidélité ▶ Parrainage ▶ Réservation
Métriques	<ul style="list-style-type: none"> ▶ Nombre codes promos utilisés ▶ Nouveaux comptes ▶ <u>Nombre ventes</u> 	<ul style="list-style-type: none"> ▶ Consultation nouvelles pages ▶ Nouveaux comptes ▶ <u>Nombre ventes</u> 	<ul style="list-style-type: none"> ▶ Nombre ventes en promos combinées ▶ Nombre vente en cross-selling ▶ <u>Nombre ventes</u> 	<ul style="list-style-type: none"> ▶ Participants vente-privée ▶ Ventes en vente privée ▶ Participants pré-lancement ▶ Codes utilisés ▶ <u>Nombre ventes</u> 	<ul style="list-style-type: none"> ▶ Nouveaux comptes ▶ Coupons utilisés ▶ Nombre réservations ▶ <u>Nombre ventes</u>

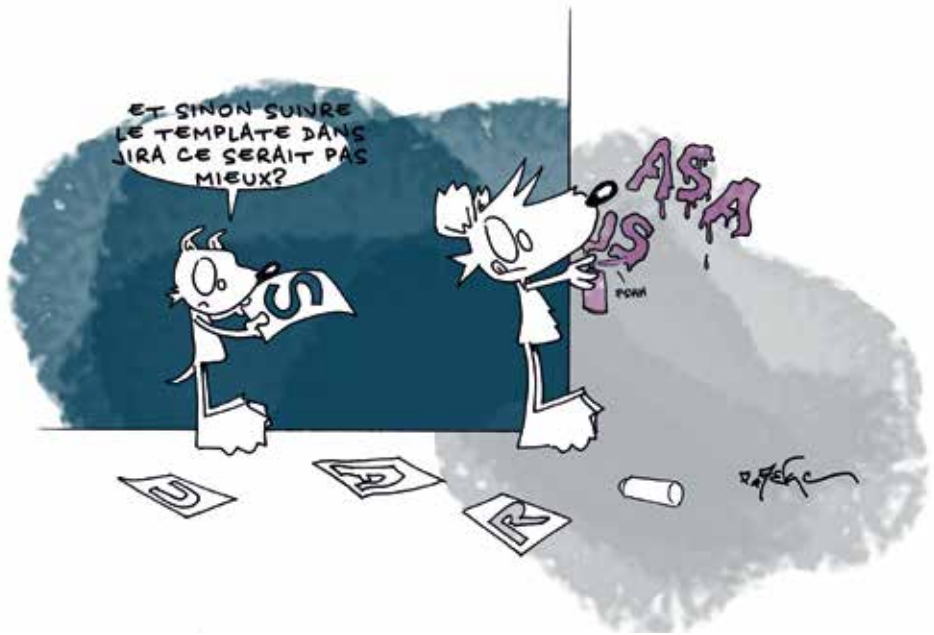
La question qui se posera alors naturellement est : « Quand livre-t-on ? ». Cette question pourrait faire l'objet d'une règle à part entière, mais l'idée principale est la suivante : il est impossible de prévoir une date de livraison précise tant que le Product Owner n'a pas une idée de la vélocité de son équipe. Une première date pourra être estimée après deux ou trois sprints et affinée au fur et à mesure que les fonctionnalités associées seront mieux définies.

L'après roadmap

La roadmap n'est pas gravée dans le marbre, bien au contraire. Elle a vocation à évoluer avec votre produit. En effet, le suivi des métriques (qui sera abordé plus précisément dans la troisième partie du livre) peut et doit influencer sur la roadmap. N'hésitez donc pas à régulièrement faire le point sur votre impact map pour vous assurer que vous avancez dans la bonne direction et faites évoluer votre roadmap si nécessaire.

Règle n°5 :

RÉDIGEZ DE BELLES USER STORIES



Structurez vos user stories

Sur ce sujet, l'idée à garder en tête est :

« Le seul bon format d'une user story est celui qui fonctionne avec votre équipe ».

C'est normalement l'équipe qui sait ce dont elle a besoin pour commencer à travailler. C'est elle qui va pouvoir dire si une user story est prête ou non, et ce quel que soit son formalisme.

Cependant, il existe une structure générique des user stories que vous pouvez utiliser comme modèle ou simplement comme source d'inspiration.

Généralement, une user story se compose :

- d'un titre : « Client détenteur d'une carte VISA règle sa commande ».
- d'une phrase narrative le plus souvent 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 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 : « Contrôle à effectuer sur le format de carte ».

Dans certaines équipes, la user story réduite à son stricte minimum suffira, dans d'autres, la user story ressemblera à une mini spécification. En la matière, nous constatons régulièrement que la longueur d'une user story peut refléter la distance qui sépare l'équipe de son Product Owner. Néanmoins, toute user story doit dans son ensemble respecter certains critères d'éligibilité. Il est possible d'utiliser le framework **INVEST** pour juger de la qualité d'une user story.

Selon ce framework, une bonne user story est :

- I** – Indépendante : elle doit se suffire à elle-même, car les dépendances avec d'autres user stories induisent des problématiques de testabilité et de planification.
- N** – Négociable : elle doit amener à la discussion et peut-être modifiée jusqu'à son inclusion dans une itération.
- V** – Valeur : elle doit apporter de la valeur à l'utilisateur final. La notion de valeur étant toujours difficile à évaluer, la user story doit être exprimée avec une vision de l'objectif recherché pour l'utilisateur.
- E** – Estimable : elle doit être suffisamment claire et comprise par l'équipe pour que celle-ci soit en capacité de l'estimer.
- S** – Small : elle doit être de taille assez petite pour prioriser de façon sûre et éviter les effets tunnels. Essayez donc au maximum de découper finement vos user stories.
- T** – Testable : la user story doit raconter une histoire, dont les tests découlent de façon évidente.

Attention ! Dans la pratique, il est très compliqué d'avoir un backlog totalement INVEST. Par exemple, il existera nécessairement des stories dépendantes entre elles que vous serez obligés de traiter les unes à la suite des autres. Cette

méthode permet surtout au Product Owner de questionner son découpage et le contenu de ses user stories. C'est donc d'avantage une ligne de conduite qu'une règle immuable.

Une bonne user story ne doit pas présupposer de la solution. Cette dernière sera construite par le Product Owner, l'équipe, le client au travers d'un dialogue autour de la compréhension du besoin et des options possibles.

De plus,

- n'hésitez pas à abuser des maquettes, schémas, mock-up, etc. : une image vaut mille mots.
- ayez en tête qu'une user story (indépendamment de son format) est avant tout une histoire qui se raconte, crée la discussion et amène l'équipe à confirmer sa compréhension du besoin.

Pour finir, la vraie question à se poser est : qu'est-ce qui fonctionne pour notre équipe ? Dans la mesure du possible, évitez le dogme, écrivez les user stories dont l'équipe a besoin pour développer, ni plus, ni moins, et demandez régulièrement à l'équipe ce qui peut être amélioré.

Mettez en place des critères d'acceptation : atelier "three amigos"

Les critères d'acceptation permettent de valider une user story. Ils vont guider les développeurs et testeurs pendant son développement ; il est donc important que ces tests soient partagés et compris par toute l'équipe. L'atelier "three amigos" aide à l'écriture des critères d'acceptation sous la forme de BDD (Behaviour Driven Development).

Comment cela fonctionne ?

Un atelier "three amigos" réunit le Product Owner, un développeur et un testeur (s'il est différent du Product Owner). La participation d'un testeur n'est pas un impératif, mais peut être utile pour s'assurer que toutes les parties prenantes de l'intégration sont alignées sur les résultats attendus.

Avant l'atelier, le Product Owner écrit les exigences des différentes user stories qui seront traitées et les fait relire aux autres participants. Ainsi, le développeur et le testeur arriveront avec une liste de questions déjà préparée.

Une fois que les éléments devant être développés sont bien compris par tout le monde, on reprend les exigences et liste ce qui devra être testé. Il n'est pas utile d'avoir un plan de test global, il faut juste se concentrer sur les principaux scénarios et la manière de les tester (test manuel, test unitaire, etc.).

Ce qui sort de l'atelier

Ce qui résulte de cet atelier est une liste de tests d'acceptation (souvent associés aux pratiques du BDD). Ils peuvent être écrits sous la forme :

- GIVEN... (un contexte)
- WHEN... (l'utilisateur effectue certaines actions)
- THEN... (ce que l'on observe)

Cette mise en forme sera faite après l'atelier, généralement par le testeur ou le Product Owner, avant d'être revue par tout le monde. La syntaxe utilisée, appelée désormais langage Gherkin, est de plus en plus reconnue par des frameworks (tels que `Behave` ou `Cucumber`) permettant d'automatiser ces tests fonctionnels.

Dans le cas d'une utilisation avec ce type de framework, il vous faudra être vigilant sur le niveau de lecture : il ne s'agit pas d'écrire un test générique, mais bien de l'exprimer avec des données précises.

En reprenant la user story proposée plus haut, un des cas de test pourrait s'exprimer ainsi :

Étant donné un utilisateur avec une carte de n°1234567890, valide jusqu'en 01/16 et de code sécurité 123,

Quand il saisit la carte 1234567890

Et la date de validité 01/16

Et le code de sécurité 123,

Alors, le paiement est accepté

Et il est redirigé vers la page de confirmation du paiement.

La liste des tests d'acceptance n'est pas le seul point bénéfique de l'atelier, bien au contraire. La vision de ce qui va être développé est maintenant commune et bien partagée par tout le monde (testeurs compris). Les risques d'aller-retour au sein de l'équipe sont alors considérablement réduits.

Attention à ne pas en abuser !

Il est important de faire preuve de bon sens avant de proposer cet atelier. Il n'est pas nécessaire de réunir trois personnes pour définir les tests d'acceptance d'une fonctionnalité basique et encore moins d'une correction d'anomalie (on sait déjà ce qui ne fonctionne pas !). Il est en revanche crucial, au moins au début d'un projet, de mettre d'accord les différentes personnes impliquées dans la réalisation d'une story sur la façon d'écrire cette dernière. Cela limitera a posteriori les incompréhensions entre ces acteurs.

Il ne faut pas perdre de vue que l'objectif est avant tout de sortir son produit rapidement... et non de générer de la documentation à tout va !

Règle n°6 :

PRIORISEZ VOTRE BACKLOG



Une fois les user stories plus ou moins détaillées dans le backlog, il s'agit ensuite de les prioriser dans les sprints.

Le Product Owner doit alors se demander par quoi commencer et pourquoi une user story est plus prioritaire qu'une autre.

L'objectif du Product Owner est de délivrer en priorité ce qu'il juge avoir le plus de valeur pour ses utilisateurs et/ou son entreprise. À l'échelle d'une user story, nous retrouvons ce principe de valeur dans la section « bénéfice attendu ». Néanmoins cette notion est souvent trop vague ou tellement évidente qu'elle ne permet pas une priorisation efficace.

Pour être pertinent, le Product Owner doit prendre en compte d'autres critères et doit élargir son panel d'outils de priorisation. Nous vous proposons dans cette règle de balayer trois outils de priorisation : MoSCoW, Kano et le story mapping.

MoSCoW

Le méthode de MoSCoW permet de prioriser les user stories à moyen terme selon les critères suivants :

M – Must have : doit être réalisée.

S – Should have : devrait être réalisée si possible.

C – Could have : pourrait être réalisée s'il n'y a pas d'impact sur les autres tâches en cours.

W – Won't have : ne sera pas réalisée tout de suite mais serait souhaitable pour une version ultérieure.

Une priorisation par la technique de MoSCoW peut être un bon point de départ, mais c'est une approche qui a certaines limites. Les participants à l'atelier risquent de tomber dans le piège du « tout est important », en raison de leurs expériences des expressions de besoins classiques et du risque – réel ou perçu – que les stories qualifiées comme “Could have” ou “Won't have” ne soient jamais réalisées. Avec cette façon de penser, une bonne réalisation d'un MoSCoW demande un changement d'état d'esprit important et difficile à obtenir. Il est de la responsabilité du Product Owner d'animer ces ateliers en véhiculant le bon message.

Kano

Cette méthode, créée en 1984 par Noriaki Kano part du constat de l'asymétrie des sentiments de satisfaction et d'insatisfaction qu'un utilisateur peut avoir face à un produit. Un utilisateur peut être moyennement satisfait de la présence d'une fonctionnalité alors que son absence peut être un motif de grande insatisfaction.

Si votre backlog est très fourni, le modèle de Kano permet une priorisation au niveau des grandes fonctionnalités de votre produit. Un atelier de priorisation avec la méthode de Kano se déroule en quatre étapes :

Étape 1 : identification des fonctionnalités.

Étape 2 : consultation des parties prenantes qui doivent répondre aux questions suivantes :

- Le produit possède cette fonctionnalité, qu'en pensez-vous ? (forme fonctionnelle)
- Le produit ne possède pas cette fonctionnalité, qu'en pensez-vous ? (forme dysfonctionnelle)

Avec comme réponses possibles :

- « Aime » (J'aimerais ça)
- « Attend » (Je m'attends à ce qu'il en soit ainsi)
- « Neutre » (Cela m'est égal)
- « Vit avec » (Je l'accepte)
- « N'aime pas » (Je n'aimerais pas ça)

O > Obligatoire/Essentielle

L > Linéaire

E > Excitante

C > Contradictoire

Q > Questionnable

I > Indifférent

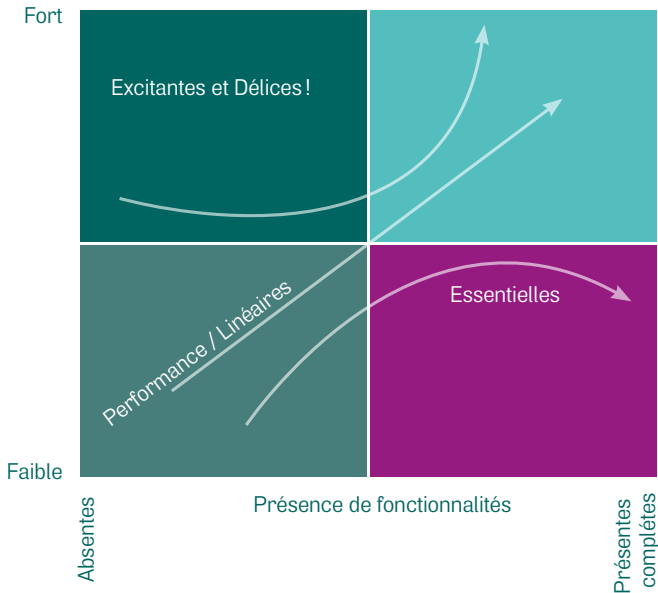
		QUESTION SOUS FORME DYSFONCTIONNELLE				
		Aime	Attend	Neutre	Vit avec	N'aime pas
QUESTION SOUS FORME FONCTIONNELLE	Aime	Q	E	E	E	L
	Attend	C	I	I	I	O
	Neutre	C	I	I	I	O
	Vit avec	C	I	I	I	O
	N'aime pas	C	C	C	C	Q

Étape 3 : analyse des résultats (comparaison entre la forme fonctionnelle et dysfonctionnelle) et identification du type de fonctionnalité :

- Fonctionnalités obligatoires ou essentielles (O) : ce sont les bases de votre produit, les fonctionnalités incontournables.
- Fonctionnalités linéaires (L) : l'addition de ces fonctionnalités permet d'augmenter la valeur de votre produit.
- Fonctionnalités excitantes (E) : le « petit plus » de votre produit par rapport à un autre.

Étape finale : visualisation sous forme de diagramme.

MODÈLE DE KANO



Cette approche est intéressante car elle se fonde sur une perception du client et elle permet de mettre en évidence des attentes parfois non exprimées.

Story mapping

Le story mapping a été largement abordé dans la règle 4 de ce livre. Il s'agit d'un excellent outil de priorisation par cohérence fonctionnelle. Une story map est une véritable cartographie du produit qui peut prendre des formes très variées, mais elle permet dans sa forme classique de prioriser entre elles des fonctionnalités de haut niveau.

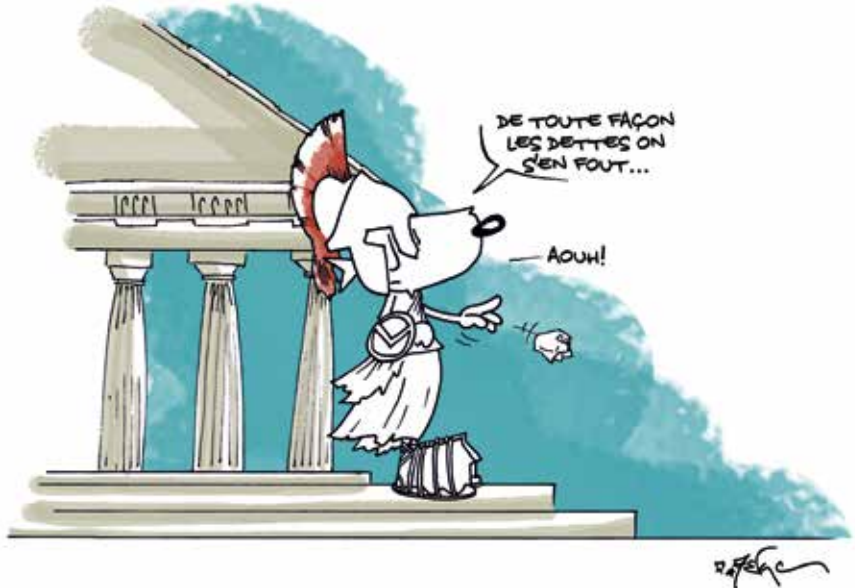
Cet exercice aboutit à une représentation différente de celle du backlog puisqu'elle met en avant à la fois des workflows de user stories et des dépendances entre des user stories à forte valeur ajoutée et d'autres non prioritaires ou même techniques.

En outre, cette représentation à d'autant plus de valeur qu'elle est le résultat de discussions et de débats entre les parties prenantes pour aboutir à une vision unique et partagée.

En résumé, les clés d'une bonne priorisation sont de prendre en compte différents critères (et non il n'y a pas que la valeur qui compte !) et d'utiliser le management visuel afin de définir collectivement les priorités.

Règle n°7 :

PRÉOCCUPEZ-VOUS DE LA DETTE TECHNIQUE



Commençons par le commencement : qu'est-ce que la dette technique ?

La dette technique est une métaphore du développement logiciel inventée par Ward Cunningham. Il s'inspire du concept existant de dette financière et l'applique au domaine du développement logiciel.

L'analogie faite par Cunningham est la suivante : les mensualités liées à un emprunt servent à rembourser une part du capital et des intérêts. Si les remboursements sur le capital ne sont pas réguliers, les intérêts, directement calculés

sur ce dernier, demeureront importants. Ainsi dans les projets logiciels, le code endetté de nos applications correspond au capital et les bugs représentent les intérêts. Dès lors que nous ajoutons de nouvelles fonctionnalités, le capital augmente et génère davantage de bugs et de maintenance.

La dette technique représente donc des parties de code non utilisées, ou dans lesquelles il est difficile d'effectuer des modifications et des évolutions.

La dette technique est inévitable... mais encore faut-il savoir l'identifier, la catégoriser et l'expliquer

Pour ce faire nous allons utiliser le Technical Debt Quadrant de Martin Fowler de Thoughtworks.

Le Technical Debt Quadrant catégorise la dette technique en quatre types bien distincts :

TECHNICAL DEBT QUADRANT

EXCESSIVE ET IMPRUDENTE

Dette produite par des équipes qui décident délibérément de faire du "quick and dirty" parce qu'elles ne peuvent prendre le temps nécessaire pour concevoir leur code proprement.

Ex: site d'e-commerce qui réalise une grande partie de son CA pendant la période de Noël. Les fonctionnalités doivent sortir avant cette période.

MODÉRÉE ET PRUDENTE

Une dette complètement délibérée pour livrer à temps une version du produit ne vaut peut-être pas la peine d'être remboursée si les intérêts sont suffisamment petits.

Ex : une partie de code rarement modifiée.

VOLONTAIRE

Dette souvent due à des équipes ignorant les pratiques de conception et d'architecture les plus basiques.

Ex : startups pour lesquelles il est plus important de valider une idée ou d'être le premier à se lancer sur un marché que d'écrire du code de qualité.

Dette technique existant dans les équipes mêmes les plus expérimentées car inévitable. Souvent ce n'est qu'à la fin d'un projet que l'on se rend réellement compte de ce que la conception aurait dû être, même si le code et l'architecture mis en place sont très bons et très bien pensés.

INVOLONTAIRE

Pourquoi le Product Owner doit-il s'en préoccuper ?

Afin de pouvoir faire évoluer son produit dans le temps et ne pas être bloqué par une maintenance beaucoup trop importante, il est important que le Product Owner veille à ce que la dette technique de son produit soit maîtrisée. Vu sous cet angle : Non, un code propre et maintenable n'est pas une perte de temps. En tant que Product Owner, vous devez être le meilleur avocat de cette philosophie vis-à-vis de votre équipe et de l'extérieur.

La dette technique concerne à la fois la maintenance, l'évolutivité et la fiabilité de votre produit :

- **la maintenance** : qu'elle soit corrective (c'est-à-dire qu'elle consiste à corriger les défauts et les bugs liés à une application) ou adaptative (c'est-à-dire qu'elle consiste à adapter une application afin que celle-ci continue à fonctionner sur des versions plus récentes des briques techniques sur lesquelles elle s'appuie), la maintenance d'une application ou d'un logiciel est au coeur de votre travail et touche directement les équipes de développement. Plus cette maintenance est conséquente, plus votre équipe de développement passera ses sprints à corriger des bugs – et donc augmentera les coûts et impactera le budget – et moins elle livrera de fonctionnalités.
- **l'évolutivité d'un produit** : l'essence même de notre métier de Product Owner est de faire évoluer notre produit notamment en proposant de nouvelles fonctionnalités aux utilisateurs. Un code « endetté » nécessitera des interventions plus longues et difficiles pour développer de nouvelles fonctionnalités.
- **la fiabilité** : vous n'êtes pas sans savoir qu'un utilisateur qui est confronté à un logiciel ou une application instable et dont le comportement est erratique, reviendra peut-être une deuxième fois, mais jamais une troisième.

Faire un produit qui ne nécessite pas de maintenance, dont les évolutions sont toujours faciles et rapides à mettre en oeuvre et qui est d'une fiabilité sans faille est très utopique. Le travail du Product Owner consiste à trouver le bon compromis entre la maximisation de l'évolutivité, la fiabilité des produits et la

minimisation du temps consacré à la maintenance. La tentation de baisser le niveau de qualité des fonctionnalités afin de les livrer dans les délais exigés existera toujours. Reste à garder un niveau de fiabilité et de qualité suffisant.

Quelles pratiques pour diminuer la dette technique ?

Les questions que vous vous posez certainement maintenant sont :

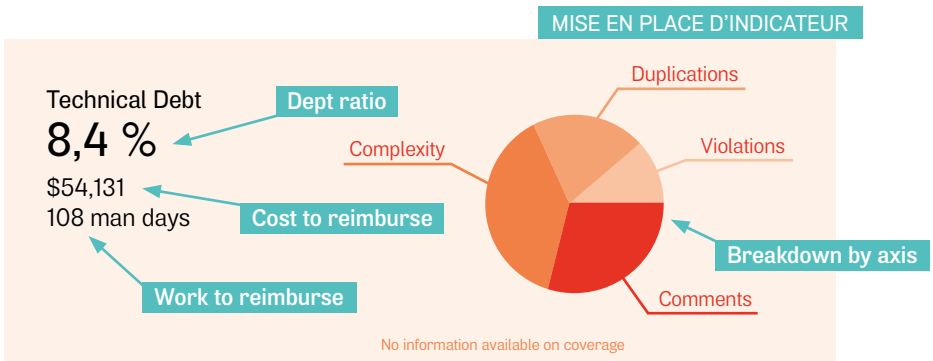
- Comment faire en sorte de produire le moins de dette technique tout en continuant à livrer vos fonctionnalités dans les temps ?
- Comment faire pour vivre avec une dette technique existante en essayant tout de même de la minimiser ?

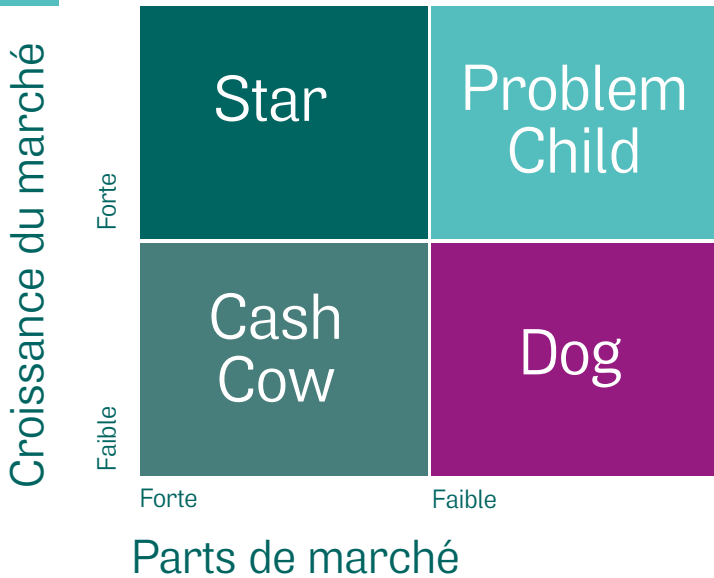
Pour inverser la tendance, il faut savoir la mesurer

Avant de s'attaquer à la dette, il faut déjà maîtriser le problème. Mettre en place des indicateurs peut être un bon départ. En tant que Product Owner, montrer un intérêt pour le sujet est un signal fort.

Une approche pragmatique consiste à commencer par mettre en place des outils d'inspection comme SonarQube et calculer un indice dédié à la dette technique (sonar propose par exemple un plugin d'indicateur de dette technique).

Même si la pertinence de cette valeur est toujours discutable, ce qui importe est que nous avons maintenant un référentiel. Nous pouvons donc étudier l'impact de nos actions !





Prioriser

Vous souvenez-vous de la matrice BCG (matrice Boston Consulting Group) ? Certes, elle est un peu “has been”, mais dans ce cas précis, elle peut vous être très utile.

Pour rappel, cette matrice nous permet de catégoriser nos produits ou applications en Stars, Cash cows, Dogs et Problem children. Pour vous attaquer à leur dette technique, vous appliquerez les règles suivantes :

- **Stars** : vous vous attellerez à supprimer le moindre bout de dette technique qui puisse exister ! Ces applications vous rapportent de l’argent, beaucoup d’argent.
- **Cash cows** : vous vous efforcerez de réduire la dette technique de façon à ce que l’application reste maintenable.
- **Dogs** : vous ne vous attaquerez pas à leur dette technique. De toute façon, ce sont des produits que vous comptez ignorer petit à petit, donc autant ignorer leur dette technique.

- **Problem children** : vous ferez en sorte de minimiser toute dette technique sur ces applications. Prévention, prévention. Ces applications sont vos bébés et potentiellement vos futures stars. En prévision d'une explosion de leur potentiel, vous les vaccinerez contre toute dette technique.

Valoriser les profils “craftsman”

Un profil de développeur est aujourd'hui particulièrement apprécié dans les équipes de développement, il s'agit du craftsman. Le mouvement software craftsmanship regroupe des développeurs qui militent pour une bonne connaissance des techniques de développement. Les craftsmen portent une attention particulière à la qualité et font preuve de pragmatisme dans leur travail.

Les craftsmen sont adeptes de la Boy Scout Rule appliquée au code “Leave code in a better state” (Robert C. Martin). Elle permettra à votre équipe de s'attaquer à la dette technique quasi gratuitement. Le principe est simple : à chaque fois que quelqu'un touche un bout de code, il devra le laisser dans un état meilleur. Une simple question d'amélioration continue !

“Definition of Done” et les bonnes pratiques de développement

Il est nécessaire de s'accorder avec votre Scrum Master sur un processus et une “definition of done” qui atteste que pour chaque user story livrée les bonnes pratiques de développement ont été respectées : lisibilité du code, tests unitaires et fonctionnels, refactoring, code review et pair programming entre autres.

Comment s'attaquer à la dette dès demain matin ?

- Faites la revue des indicateurs à chaque rétrospective, la dette technique sera ainsi un sujet régulièrement étudié par l'équipe.
- Indexez les estimations sur l'indicateur de dette technique ci-dessus afin de faire prendre conscience de cette dette. Ainsi, l'estimation du coût d'une fonctionnalité sera plus importante si le code est fortement endetté.

- Organisez des journées de nettoyage. Si l'équipe a du mal à maîtriser sa dette au jour le jour ou si le code est endetté de longue date, il peut être intéressant de faire cet investissement.
- Ne dédiez jamais tout un sprint au refactoring. Celui-ci consiste à retravailler le code source d'un logiciel sans ajouter de fonctionnalités ni corriger de bugs, mais en améliorant sa lisibilité pour simplifier sa maintenance. Encouragez l'équipe à proposer des refactorings sur un périmètre précis et dont la valeur est prouvée. L'équipe devra découper les sujets pour en maîtriser les potentielles dérives, comme pour les user stories !

Règle n°8 :

PLUSIEURS PRODUCT OWNERS POUR UNE ÉQUIPE DE DÉVELOPPEMENT, COMMENT PRIORISER ?



« Un backlog, une équipe, un Product Owner ! ». Voilà la bonne façon de s'organiser. Product Owner, c'est un métier à temps plein, entre la gestion du backlog, l'alimentation de l'équipe, répondre à des sollicitations...

Dans certains cas, il arrive qu'un Product Owner soit obligé de partager avec d'autres Product Owners une même équipe de développement. Cela peut venir de la culture de l'entreprise ou de contraintes organisationnelles. Un tel mode de fonctionnement est un anti-pattern agile. Vous êtes dans cette situation ? Nous allons vous expliquer comment faire au mieux.

Cas numéro 1 :

une équipe, plusieurs projets, plusieurs Product Owners

Vous vous retrouvez au sein d'une équipe alimentée par plusieurs Product Owners, pour traiter plusieurs projets en parallèle ; il s'agit, de facto, d'un centre de services. Dans ce type d'organisation, l'équipe de développement risque de devoir prioriser elle-même son backlog afin de tenir ses engagements auprès de toutes les parties prenantes.

Cas numéro 2 :

une équipe, un projet, plusieurs Product Owners

Cette situation est typique d'une équipe dont la taille trop importante la rend difficile à alimenter par un seul Product Owner. Cela peut également arriver dans le cas où des expertises métier très pointues sont nécessaires. Il faut alors identifier un Product Owner par domaine de compétences. Les risques inhérents à ce mode organisationnel sont : la difficulté à alimenter l'équipe de développement, la complexification de la priorisation et le moindre alignement.

Conséquences

L'équipe devient alors un goulet d'étranglement, et les conséquences sont connues : problèmes de qualité, usure des personnes, tensions, etc.

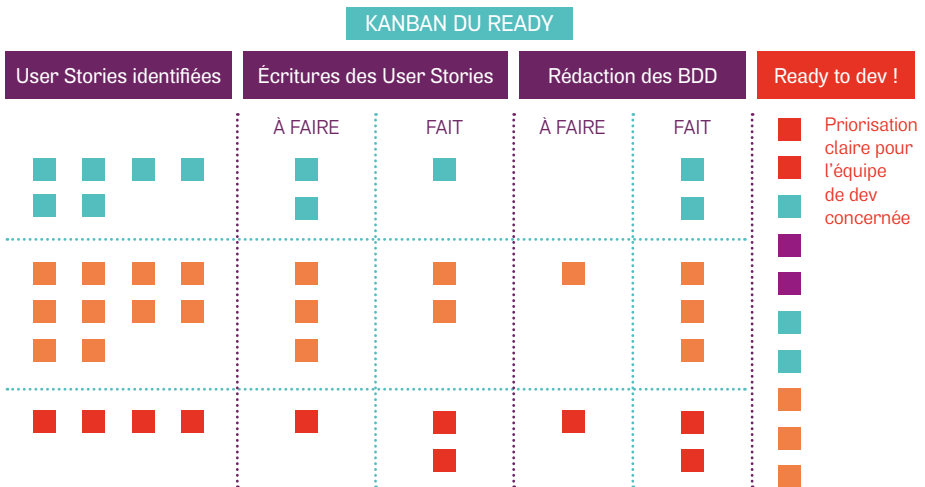
Malheureusement, les leviers nécessaires au changement organisationnel sont rarement entre les mains des Product Owners ou de l'équipe de développement, il faut donc faire avec. Comment gérer une telle situation ? Voici quelques clés pour s'en sortir.

Une solution :

mettre en place votre Kanban du Ready

Sans avoir à révolutionner votre mode de fonctionnement, une solution à mettre en oeuvre consiste à rationaliser et rendre visible la situation actuelle dans le but de provoquer une prise de conscience et d'aboutir à des améliorations. Vous êtes dans la situation d'un centre de services, prenez cet état de fait en compte. Votre équipe de développement a une capacité à faire, et celle-ci n'est pas extensible. Souvent, chaque Product Owner a sa propre roadmap, ses propres échéances et ses propres engagements. Cette situation peut vite devenir problématique. Comment l'équipe de développement obtient-elle une priorisation indiscutable des éléments de travail ? Par quoi commencer ?

Dans le cas d'un centre de services, il faut voir l'équipe comme un système et penser en flux. Bref, adopter une logique Kanban. Que vous vous trouviez dans l'un ou l'autre des deux cas décrits plus haut, la colonne du management visuel où l'équipe de développement s'alimente doit être unique, correctement priorisée grâce au travail de l'ensemble des Product Owners concernés. Cela signifie que l'ultime étape du travail des Product Owners est une phase d'alignement sur les priorités pour le développement. Ces priorités sont explicitées dans la colonne Ready to dev (Prêt à développer) du Kanban.



En plus de l'alignement sur les priorités de développement, ce Kanban du Ready a deux grandes vertus :

- Les éléments de travail pour l'équipe de développement ont une forme cohérente.
- L'équipe de développement dispose d'une vision exhaustive de l'activité des Product Owners et un aperçu de ce qui les attend très prochainement.

Dans le schéma ci-dessus, chaque Product Owner voit son activité affichée dans une ligne de nage du Kanban global. Une fois qu'un nombre suffisant de user stories sont matures pour lui et ses condisciples, il est possible de déclencher une séance de présentation des user stories à l'équipe, de collaborer entre Product Owners et de déterminer le bon ordre de priorité, non ambigu, des user stories à envoyer en développement. Les Product Owners doivent s'entendre là-dessus !

C'est également le moyen d'adopter une approche de juste à temps pour la maturation des user stories : l'équipe de développement a une capacité fixe, un débit. Il faut prendre en compte ce débit jusque dans l'expression des besoins, la planification des ateliers métier, la sollicitation éventuelle des équipes de test, etc. L'utilisation de limites sur les activités de votre système va permettre le passage d'un flux poussé vers un flux tiré. Le travail d'alimentation des Product Owners va donc se caler sur la capacité d'absorption des développeurs. En effet, pourquoi écrire, discuter, estimer, bref, perdre du temps sur des user stories que l'équipe n'a pas la capacité d'absorber ? Les Product Owners au « chômage technique » seront alors plus utiles à aider l'équipe à tester ce qui est commencé ou d'autres Product Owners à finir de rédiger les user stories les plus importantes. On appelle cela le "swarming".

Selon votre contexte et la culture de la société dans laquelle vous évoluez, une équipe de Product Owners peut utiliser cette approche pour s'aligner sur les priorités et obtenir cette fameuse colonne unique et priorisée "Ready to dev" - « Prêt à être développé ». Parfois, le consensus est facilement atteint.

Malheureusement, nous avons parfois besoin de prioriser et donc de quantifier l'intérêt de telle user story par rapport à telle autre. Dans ce cas, manipuler la valeur métier ne suffit pas – cette valeur est de toute façon souvent incomprise ou mal utilisée. Une bonne approche est alors d'utiliser une forme minimale de coût du délai. Combien me rapporterait ou me ferait économiser une user story si elle était livrée ?

Quatre classes de services sont généralement utilisées pour classer les éléments de travail :

- Urgence : une user story qui rapporte ou fait économiser immédiatement de l'argent à l'entreprise ; chaque jour qui s'écoule constitue donc un manque à gagner.
- Date fixe : une user story qui doit être livrée à une date donnée, typiquement une obligation légale. Au delà de cette date, chaque jour écoulé risque de faire perdre une somme importante (une amende par exemple).
- Standard : une user story classique qui a une valeur business, chaque jour écoulé vous éloigne d'un gain potentiel.
- Intangible : typiquement la dette technique, l'intangible ne presque coûte rien jusqu'au moment où il est trop tard, la bonne pratique est de gérer un peu d'intangible tout le temps.

Cette technique de priorisation est un moyen simple pour challenger la valeur d'éléments de travail, notamment pour des lignes produit ou projet de prime abord « incomparables ».

Dans tous les cas, l'esprit agile doit prévaloir : la résolution des questions liées à ce mode de fonctionnement est grandement facilitée par un esprit d'équipe à toute épreuve et le partage de la vision. La mise en place d'un Kanban du Ready rendra le problème visible mais ne remplacera jamais l'échange entre les Product Owners. Une cérémonie stricte de priorisation calée sur un rythme régulier est indispensable. La mise en place d'une communauté de pratique des Product Owners facilitera l'alignement, la transparence et renforcera l'esprit d'équipe.

Règle n°9 :

ADOPTEZ LA BONNE POSTURE



D'une manière générale, les Product Owners embrassent les valeurs de l'agilité, dont la collaboration et l'entraide. Ils n'ont pas de lien hiérarchique avec les développeurs, et sont des membres de l'équipe à part entière plutôt que les « chefs » des équipes de développement. À ce titre, ils doivent adopter une posture toute particulière que nous allons détailler dans cette règle.

Les qualités du Product Owner

Il est polyglotte

Le Product Owner est au carrefour des univers métier et technique. Il doit donc porter son attention dans ces deux directions à la fois.

Une des principales qualités d'un Product Owner est d'être polyglotte : il sait parler à des commerciaux, des équipes marketing, des équipes techniques et... des experts métiers.

Il doit être capable de transmettre et véhiculer des informations, à tout moment, en adaptant son discours à son interlocuteur.

Ainsi, il est en mesure de répondre aux questions métier que peut se poser l'équipe de développement, et à l'inverse de pouvoir fournir des points d'avancement ou des alertes aux métiers sur les réalisations en cours, sans rentrer dans les considérations techniques.

Il aime communiquer, communiquer et communiquer

Cette qualité nous semble un pré-requis à la réussite de la mission principale du Product Owner à savoir « porter » la vision produit auprès des membres de l'équipe. Le Product Owner explique la valeur métier du besoin, et pas seulement le « quoi », afin que les développeurs puissent se l'approprier.

Il est curieux et un peu geek

Sans demander au Product Owner de réellement développer le produit, il se doit d'être curieux et de savoir comment est construit son produit (ex : Quelles API sont utilisées pour livrer cette fonctionnalité ?). Le Product Owner doit être en mesure d'expliquer aux métiers les compromis et les décisions techniques prises par l'équipe de développement.

Il est autonome sur les outils et les petites tâches techniques. Il est très agréable pour une équipe de développement de ne pas être sollicitée par un Product Owner qui vient la voir tous les jours pour lui demander de lui changer son mot de passe, de manipuler tel ou tel jeu de données ou pour lui demander pour la n-ème fois l'URL de la plate-forme de pré-production.

Comme un bon scout, il est toujours prêt !

Le Product Owner a conscience que le développement produit n'est pas un processus gravé dans le marbre et nécessite en permanence des ajustements. Il travaille donc en collaboration avec l'équipe lorsque des changements doivent être faits ou que des obstacles sont rencontrés. En ce sens, il est « agile » car il sait adapter son backlog en fonction des circonstances.

L'équipe accordera plus facilement sa confiance à un Product Owner si elle sent qu'il peut s'adapter rapidement et qu'il est capable de proposer des alternatives.

Et pour finir, il est fun !

Un Product Owner positif et amusant peut apporter une grande contribution au moral de l'équipe. Vous serez forcément confronté à des périodes difficiles et dans ces moments là, une dose d'humour peut grandement aider à passer ce cap. Surtout, soyez humble et donnez à l'équipe de développement le crédit qu'elle mérite !

L'attitude vis-à-vis de l'équipe de développement

Au sein de l'équipe, même si les rôles sont importants, en tant que Product Owner vous ne devez pas être mis sur un piédestal. Vous devez montrer que vous êtes un membre à part entière de l'équipe et que vous êtes convaincu par l'approche itérative et incrémentale.

Comme membre de l'équipe, vous participez à toutes les cérémonies. Que vous soyez en Scrum ou au sein d'un système Kanban, vous êtes concerné par le planning poker, le sprint planning ou la simple présentation d'un lot de user stories prêtes à être développées, le point quotidien, la démonstration, les rétrospectives ou les ateliers de résolution de problèmes ou d'amélioration continue.

Le point quotidien

Participer au point quotidien vous permet de communiquer sur votre activité de la même manière que tout équipier. Ainsi, les développeurs auront de la visibilité sur les prochaines user stories sur lesquelles ils vont très prochainement travailler et se rendront également compte que parfois, alimenter un backlog de manière pertinente n'est pas un long fleuve tranquille.

La présentation des user stories

Il semble naturel que le Product Owner participe à la présentation des user stories, puisque celles-ci doivent inviter à la conversation entre tous les membres de l'équipe. Cependant, l'apport du Product Owner ne doit pas se limiter à répondre aux questions des développeurs sur ses user stories ; en

étant attentif aux conversations techniques qui alimentent le planning poker, le Product Owner peut déceler des signaux faibles concernant le travail à faire : telle story entraîne un travail de conception collaboratif, telle autre semble triviale ou encore le feedback des équipiers ouvre des perspectives inédites pour de futures stories...

La rétrospective

Certains Scrum Masters peuvent “interdire” l'accès à la rétrospective au Product Owner (si, si, c'est arrivé). Cet anti-pattern est bien souvent un mauvais signal. Aucune équipe agile avancée ne se tire une telle balle dans le pied : le Product Owner doit pouvoir tout entendre et tout dire. Il doit pouvoir remonter un dysfonctionnement qu'il a constaté mais également être prêt à entendre que sa manière de travailler est remise en cause.

En participant activement à toutes ces cérémonies, le Product Owner sera en mesure de remplir correctement son rôle en étant le garant du développement du bon produit, en assurant la bonne visibilité de l'avancement vis-à-vis des parties prenantes et en étant un acteur de l'amélioration continue de l'équipe.

Enfin, la résolution de problèmes est un sport collectif. Toutes les idées sont bonnes à prendre, et embarquer une personne qui n'est pas forcément technique est une marque d'ouverture d'esprit !

L'attitude vis-à-vis des parties prenantes

Donner une visibilité globale

Si vous travaillez dans une approche Kanban, vous avez rendu les étapes de maturation des user stories visibles. Même en Scrum, un management visuel de cette progression est de plus en plus adopté et a de nombreux avantages : le board renforce la cohésion de l'équipe et permet surtout au Product Owner d'organiser son travail, de planifier ses ateliers et de s'assurer que l'équipe de développement sera bien alimentée en user stories qui satisfont la Definition of Ready.

Savoir dire non

Nous ne pouvons pas clore cette règle sans parler de la posture du Product Owner vis-à-vis de ses métiers ou de son management. Si le Scrum Master se pose parfois en gardien de l'équipe, le Product Owner doit se poser en « gardien » du produit.

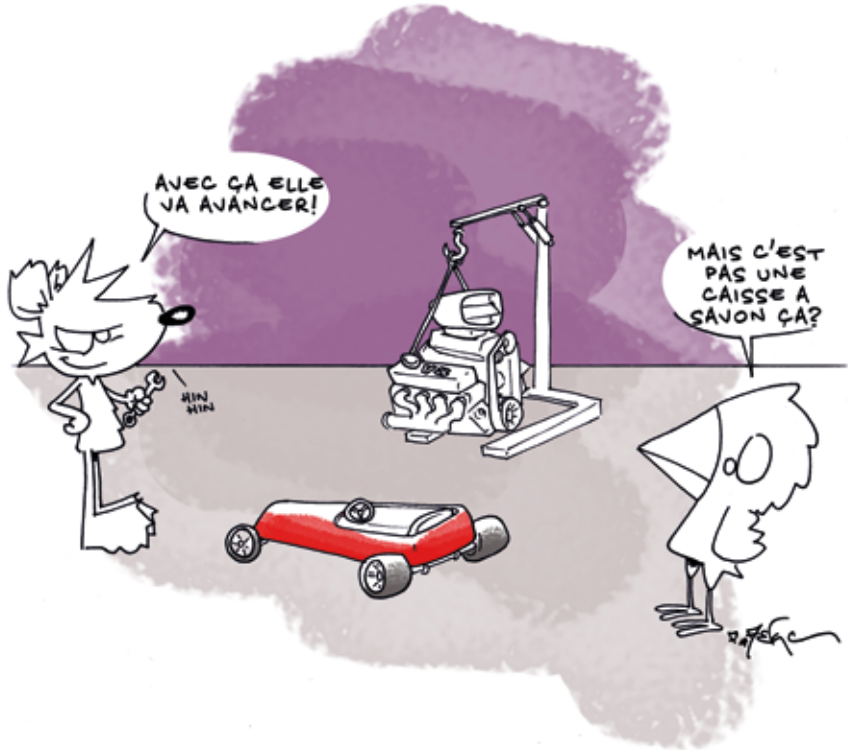
Non, tout n'est pas pour tout de suite. Non, telle fonctionnalité n'a pas forcément sa place dans cette version, voire dans le produit. Non, il n'y aura pas de miracle et la date d'atterrissage ne va pas magiquement avancer.

Ce rôle est difficile à tenir. Il faut, diplomatiquement, parfois porter des discours que vos métiers ou votre management n'ont pas envie d'entendre. Cela fait aussi partie des lourdes responsabilités du Product Owner.

Faire évoluer le produit

Maintenant que votre produit est réalisé (et lancé sur le marché), vous devez l'améliorer en continu. Les règles de cette partie vous expliquent comment vous y prendre.

Règle n°10 : ENRICHIR VS. OPTIMISER



Une fois lancé sur le marché, votre produit devient en quelque sorte « exposé » et vous voyez affluer sur votre bureau des demandes venant de toutes parts :

- Les utilisateurs vont remonter des bugs.
- Les utilisateurs vont suggérer des évolutions.
- En interne, le CEO va demander à ce qu'on ajoute une petite fonctionnalité car il en a discuté avec un ami lors d'un dîner.
- Si c'est un produit B2B, les commerciaux vont passer dans le bureau avec un ultimatum : « Si tu ne me fais pas cette fonctionnalité dans la prochaine release, je perds le deal et je ne fais pas mon quartier ».

En plus de ces demandes externes, votre métier de Product Manager vous amène à réfléchir en permanence à de nouvelles idées pour faire évoluer votre produit, que celles-ci proviennent d'une inspiration géniale ou d'une bonne pratique observée chez vos concurrents.

Vous devez alors prendre du recul vis-à-vis de ces différentes sources d'inspiration (parfois contradictoires) et vous forger une conviction quant à l'évolution de votre produit. En pratique, cet exercice peut s'avérer difficile !

Voici quelques conseils qui pourront vous aider à arbitrer entre l'envie de faire évoluer le produit avec de nouvelles fonctionnalités et la nécessité d'optimiser l'existant.

Ne vous faites pas dicter les évolutions de votre produit par vos utilisateurs

Depuis quelques années, nous entendons beaucoup parler de conception centrée sur l'utilisateur. Sur le principe, c'est évidemment une très bonne chose ! Un produit doit être imaginé et réalisé dans le but de répondre à une problématique utilisateur. Cependant, il y a parfois une confusion entre une conception centrée sur l'utilisateur et une conception guidée par l'utilisateur.

En effet, certains d'entre eux vont :

- Réclamer une fonctionnalité à cor et à cri en signifiant (de bonne foi !) qu'ils seraient prêts à payer beaucoup plus cher pour cela – mais rien n'étaye le fait qu'ils le feraient.
- Manifester un mécontentement vis-à-vis d'un problème ou d'une limitation rencontrée – alors qu'ils continuent d'utiliser le produit, voire de payer.
- Et d'une manière générale, exprimer des désirs ou frustrations qui, de toute façon, n'ont pas forcément une pertinence avérée.

Surpondérer la voix des utilisateurs par rapport aux autres sources d'inspiration peut donc vous amener à faire des évolutions non pertinentes pour votre produit.

Dans son livre “Rework”, la société Basecamp (anciennement 37 Signals) explique comment elle évite cet écueil. Les dirigeants avouent clairement ne pas prêter beaucoup d’attention aux feedbacks. Selon eux, le faire serait le meilleur moyen de construire non plus un produit mais un empilement de fonctionnalités. Le chapitre est d’ailleurs intitulé “say no by default”.

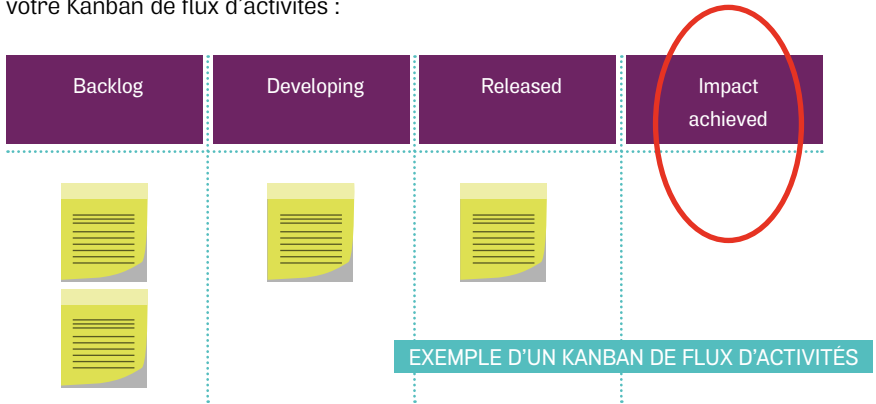
“Don’t believe that ‘customer is always right’ stuff. Let’s say you’re a chef. If enough of your customers say your food is too salty or too hot, you change it. But if a few persnickety patrons tell you to add bananas to your lasagna, you’re going to turn them down, and that’s OK. Making a few vocal customers happy isn’t worth it if it ruins the product for everyone else.”

Cette vision est un peu extrême mais Basecamp nous prouve tous les jours qu’elle peut être efficace. Ainsi, être “user centric” ne veut pas dire répondre à tous les desiderata des utilisateurs mais bien chercher en permanence à comprendre leurs problèmes. Les solutions qu’ils nous proposent ne sont que des insights dont vous devez tenir compte dans les arbitrages d’évolution de votre produit.

Validez l’impact des fonctionnalités existantes avant d’en ajouter de nouvelles

Lorsque vous lancez de nouvelles fonctionnalités, il est probable qu’elles ne répondent pas parfaitement aux besoins de vos utilisateurs (et ce, même si vous avez pris la peine de tester soigneusement vos pistes de solution en amont). Vous serez probablement amené à réaliser des ajustements en fonction des retours clients. C’est pourquoi, nous vous conseillons de vérifier qualitativement (via des entretiens) puis quantitativement (via l’analytics) la pertinence de vos nouvelles fonctionnalités avant de chercher à en ajouter de nouvelles. Vous devez ancrer cette validation « post-lancement » de vos évolutions produit dans vos processus de travail.

Pour cela, nous vous conseillons d'ajouter une colonne "impact achieved" à votre Kanban de flux d'activités :



Les objectifs chiffrés avant tout

Vous hésitez toujours entre enrichir et optimiser votre produit ? Laissez-vous guider par vos KPIs ! Chacune de vos décisions doit permettre de les améliorer.

Au-delà des indicateurs business classiques (revenu, marge, etc.), le framework AARRR offre une bonne base de départ pour cadrer vos objectifs, autour de cinq enjeux mesurés chacun par une série de KPIs :

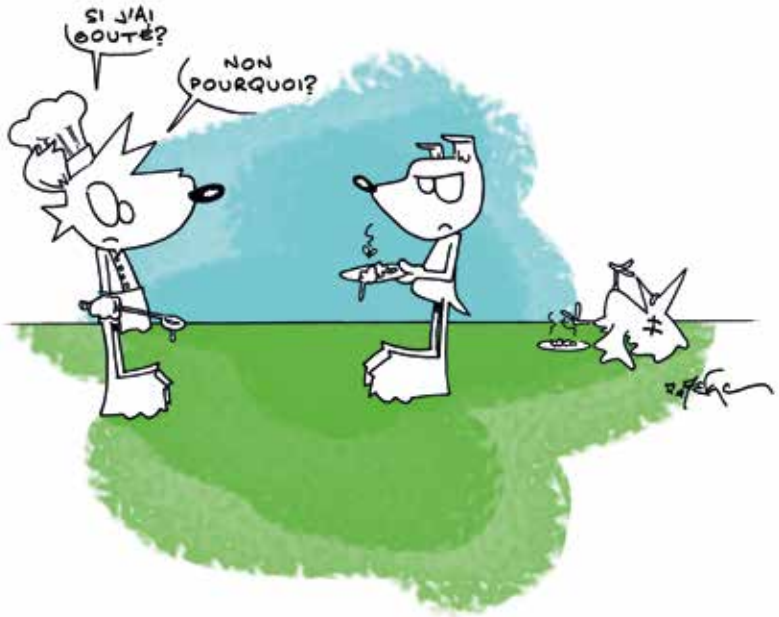
- Acquisition : faire en sorte que les utilisateurs accèdent à votre produit.
- Activation : s'assurer que vos clients utilisent vraiment le produit.
- Rétention : réussir à retenir vos utilisateurs dans le temps.
- Revenue : monétiser de votre produit.
- Referral : pousser vos utilisateurs à recommander votre produit (viralité).

Ce framework sera développé plus longuement dans la règle n°12. À ce stade, il suffit de retenir que toutes les décisions que vous prendrez, qu'il s'agisse d'introduire une nouvelle fonctionnalité ou d'optimiser un écran existant, doivent faire bouger le curseur significativement sur un de ces enjeux.

Nous vous conseillons également de mesurer régulièrement votre satisfaction client (par exemple, via le Net Promoter Score). Ceci vous amènera peut-être à modifier voire supprimer certaines fonctionnalités rejetées par une majorité d'utilisateurs. En effet, l'ajout massif de nouvelles fonctionnalités peut diluer votre proposition de valeur et générer de l'insatisfaction.

Règle n°11 :

EAT YOUR OWN DOG FOOD



À l'origine utilisée en interne chez Microsoft et Apple, cette expression résume une culture d'entreprise selon laquelle les employés doivent être les premiers utilisateurs des produits de leur société.

Voyons quels sont les bénéfices de cette pratique que nous vous recommandons vivement.

Pour comprendre les clients

En tant que Product Manager, vous devez être le tout premier utilisateur de votre produit : c'est la meilleure manière pour comprendre comment il est utilisé. Vous vous devez de connaître votre produit par coeur car vous en êtes le garant : ses forces, ses faiblesses n'ont pas de secret pour vous.

En utilisant votre propre produit, vous faites preuve d'empathie envers vos clients, et c'est en vous mettant à leur place que vous pourrez mieux comprendre leurs attentes. Cela vous permettra également de déceler de potentiels problèmes techniques ou d'ergonomie.

Un dernier avantage à cette pratique du dogfooding est de vous faire prendre de la hauteur par rapport à votre travail quotidien : vous sortez ainsi le nez de votre backlog pour avoir une vision plus holistique.

Attention cependant à la force de l'habitude : en connaissant le produit par coeur, vous risquez de ne plus avoir le détachement nécessaire pour vous mettre dans la peau d'un utilisateur « lambda ». N'hésitez pas à lire ou relire vos personas, qui vous permettront de vous détacher de votre rôle de Product Manager pour imaginer ce que peuvent ressentir ou penser tel ou tel utilisateur face à votre nouvelle fonctionnalité.

Pour avoir une image fidèle de l'avancement

Le dogfooding permet à votre équipe produit au sens large (Product Manager, développeurs, etc.) d'avoir une vision réaliste de l'état de votre produit. Les frameworks agiles (Scrum, Kanban) proposent une batterie d'indicateurs d'avancement. Cependant, rien ne remplace une démonstration concrète qui permettra à l'équipe de se situer par rapport aux objectifs.

Pour fluidifier le développement

Le dogfooding peut également être rapproché de la culture du test, chère aux agiles. En plus d'être des utilisateurs quotidiens de votre produit, vous avez le réflexe de tester les nouvelles fonctionnalités au fil de l'eau. Cela vous permettra :

- D'éviter des allers-retours entre développement et recette.
- D'avoir un rythme de livraison fréquent et régulier.

Il faut bien noter que le dogfooding est une bonne pratique mais ne remplace pas une vraie campagne de tests faite par une équipe dédiée.

Pour être les premiers ambassadeurs du produit

En poussant le concept encore un peu plus loin, il faut encourager tous les employés de votre entreprise à utiliser vos propres produits (si cela s'y prête). Cela vous permettra d'une part d'avoir du feedback d'une population non spécialiste et d'autre part de développer au sein de la société un vrai sentiment de fierté pour son travail. Vos collaborateurs seront ainsi les premiers ambassadeurs du produit ! Pousser vos commerciaux à utiliser votre produit au quotidien les rendra également plus crédibles : ils pourront par exemple effectuer des démonstrations aux clients en utilisant leur propre compte plutôt qu'un jeu de données de test. Ils seront également plus efficaces, en maîtrisant sur le bout des doigts les forces et faiblesses du produit.

Règle n°12 : GAGNEZ DE NOUVEAUX UTILISATEURS



Cette règle concerne plus particulièrement les produits SaaS B2C.

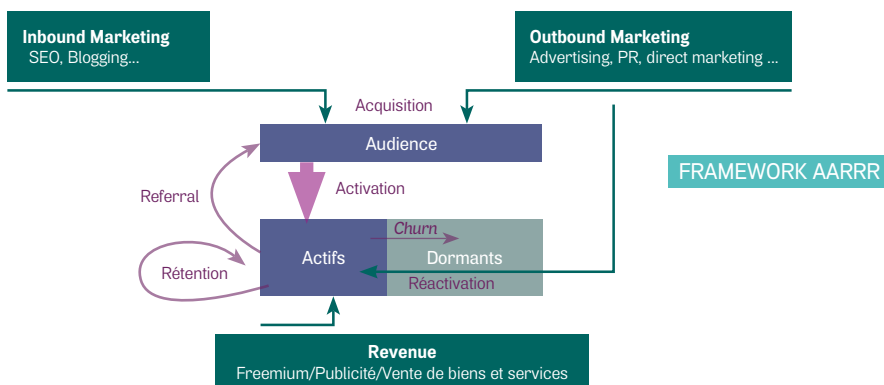
La conquête client est l'une de nos toutes premières préoccupations en tant que Product Manager, car nous souhaitons très naturellement que notre produit soit entre les mains de nombreux utilisateurs. Comment faire ? Quelles sont les étapes ?

Retour aux fondamentaux

Avant toute chose, il faut fixer des objectifs chiffrés en cohérence avec votre vision produit. En plus des indispensables objectifs financiers (de type chiffre d'affaires, taux de marge et ROI), nous vous conseillons fortement de déterminer une ambition sur l'ensemble des indicateurs clés du parcours utilisateur. Pour ce faire, vous pouvez, par exemple, utiliser le framework AARRR de Pirate Metrics. Ce framework propose cinq grandes familles de KPIs calquées sur les étapes de la vie d'un utilisateur :

- Acquisition : performance des canaux par lesquels vos utilisateurs accèdent à votre produit (Inbound ou Outbound Marketing, etc.).
- Activation : capacité à activer vos utilisateurs acquis (un utilisateur sera considéré comme activé quand il aura eu un premier usage significatif de votre produit). On parle aussi d'engagement ou d' « onboarding ». Il peut aussi s'agir de la réactivation par des campagnes marketing (push, email, sms, etc.) d'utilisateurs « dormants » c'est-à-dire ayant utilisé votre produit un temps et fini par l'abandonner.
- Rétention : capacité à retenir vos utilisateurs activés dans le temps.
- Revenu : monétisation de votre produit (par exemple, conversion d'utilisateurs gratuits en utilisateurs premium payants).
- Referral : propension de vos utilisateurs à en recruter de nouveaux par recommandation (dynamique virale, partage sur les réseaux sociaux, bouche à oreille de toute sorte, organisée par vos soins ou pas).

Visuellement, le framework s'articule de la manière suivante :



Construisez et optimisez votre “funnel”

Maintenant que nous avons pris un peu de recul sur le processus global qui nous permettra d'accéder au graal, regardons comment optimiser les phases d'acquisition, d'activation et de “referral”.

NB : Les aspects liés à la rétention sont abordés en détail dans la règle 13, « Gagner un nouvel utilisateur c'est bien, le faire revenir c'est mieux. »

Première étape : Faire arriver les prospects “chez vous”

Avec le département marketing ou le Product Marketing Manager, construisez une stratégie d'acquisition qui va vous permettre d'attirer les utilisateurs. Elle s'appuiera sur des éléments tels que :

- Votre référencement naturel ou SEO (Search Engine Optimisation).
- Vos campagnes marketing si vous disposez (ou achetez) de bases de client (e-mails par exemple).
- Vos campagnes payantes type SEM (Search Engine Marketing).
- Votre communication auprès d'acteurs influents (blogueurs, journalistes, etc.).
- Pour les applications mobiles, l'optimisation de votre visibilité sur les stores (on parle de ASO pour App Store Optimization).

Dans tous les cas, quel que soit le dispositif, ces prospects vont arriver « chez vous », sur une “landing page”. Voici quelques pistes permettant de minimiser le taux de rebond des visiteurs :

- Améliorer la performance de vos landing pages en analysant le comportement de vos prospects sur celles-ci (compréhension des messages marketing et des call to action etc.) et en pratiquant régulièrement l'A/B testing.
- Affiner le ciblage de vos campagnes en collaboration avec le marketing (ex : publicité ciblée sur Facebook).
- Se rappeler qu'un utilisateur peut venir du web, du mobile ou d'une tablette. Votre produit peut être une application, une webapp ou les deux ! Il faut donc organiser vos différents dispositifs de manière cohérente :
 - Permettre le passage de la version “desktop” à la version “mobile”

le cas échéant, ou étudier les solutions de “responsive design” (une même version de votre site s’adaptant à tout terminal de manière élastique).

- Proposer à bon escient le téléchargement de votre application suivant le terminal utilisé par votre prospect.
 - Ne pas casser de liens ! (ex : les articles de news partagés sur Facebook qui vous font arriver sur une homepage de la version mobile et vous empêchent de lire l’article).
- Essayer de se « souvenir » de ceux qui sont déjà venus, en proposant le cas échéant plusieurs versions de landing page :
 - Une version anonyme pour les utilisateurs inconnus.
 - Une version personnalisée en version « non authentifiée » : vous avez reconnu l’utilisateur mais il ne s’est pas encore authentifié ou ré-authentifié – c’est l’occasion d’adapter votre landing page pour mieux les accueillir.
 - Une version authentifiée pour les utilisateurs qui reviennent – c’est l’occasion de faciliter leur retour.

Seconde étape : Convertir vos prospects en utilisateurs inscrits

Une fois arrivés sur votre landing page, les utilisateurs vont être invités à s’inscrire à votre produit. Vous allez alors constater une fuite significative dans votre funnel. Votre rôle sera de la minimiser. Vous pourrez par exemple chercher à :

- Comprendre qui parmi vos prospects se lancent dans une inscription. En théorie, si vous les avez bien ciblés et que votre landing page est limpide, la plupart de vos prospects devrait se lancer. Si ce n’est pas le cas, revenez à l’étape suivante – votre ciblage n’est peut être pas bon ?
- Analyser les étapes correspondant aux fuites de trafic les plus importantes. En tant que Product Manager, vous aurez probablement envie de récolter un maximum d’informations sur vos utilisateurs. Gardez tout de même en tête que chaque écran supplémentaire ou chaque nouveau champ de formulaire à remplir vous fera perdre des utilisateurs par un simple effet mécanique d’évaporation : plus vous incluez d’étapes d’inscription ou complexifiez le parcours utilisateur, moins vous serez performant.

- Proposer les modes d'inscription les plus adaptés à votre cible – vous devez faciliter la vie de vos utilisateurs pour qu'ils s'inscrivent rapidement, mais sachez que nombre d'entre eux seront vraisemblablement sensibles à la manière dont vous traitez leurs données personnelles. Les "Facebook Connect" et autres API de connexion peuvent être perçus comme intrusifs. Laissez donc le choix à l'utilisateur de son mode de création de compte et mesurez ensuite ce qu'il préfère.

Troisième étape : Activer vos utilisateurs pour en faire de « vrais » utilisateurs

Voilà l'étape la plus importante, celle révélant les qualités intrinsèques de votre produit. Il va donc falloir construire habilement la première expérience clé de votre produit. Vous devez définir ce que l'utilisateur expérimentera en premier :

- Quelle fonctionnalité ou jeu de fonctionnalités constituent un tout cohérent, permettant au client de comprendre le type de problèmes ou de besoins que votre produit adresse ?
- Comment faire en sorte que ce problème ou besoin soit adressé d'une manière cohérente, agréable et pratique pour lui ?

Pour améliorer votre activation, plusieurs solutions sont possibles :

- Scénariser au mieux cette première expérience, car il s'agit d'une découverte.
- Proposer éventuellement plusieurs points d'entrée, pour laisser le choix de plusieurs premières expériences.
- Rendre la progression de cette découverte apparente (ex : votre profil est rempli à 75% sur LinkedIn).
- Gamifier la progression de la découverte (ex : déblocage de fonctionnalités au fil de l'utilisation et accumulation de points dans l'application de navigation GPS Waze).

Tout ce travail, essentiel à une activation efficace, s'organise en étroite collaboration avec l'équipe Design, et fera l'objet d'itérations perpétuelles pour optimiser la performance d'activation (identification et élimination des étapes « douloureuses » ou inutiles pour l'utilisateur).

Au-delà des analytics, vous pourrez trouver des pistes d'amélioration de votre premier parcours sur tous les canaux révélant la voix du client :

- Avis sur les “stores” ou autres forums.
- Retours faits au support.
- Entretiens en direct si vous mettez en place une démarche de type du “Customer Development”.

Et la viralité dans tout ça ?

Une fois vos problèmes de rétention réglés (cf. règle 14), souciez-vous de la viralité de votre produit ! Il s'agit de faire porter une partie de l'acquisition par vos propres utilisateurs.

Pour savoir où vous en êtes et bâtir un plan d'action autour de la viralité, nous vous conseillons de mesurer deux indicateurs, le K-Factor et le Net Promoter Score (NPS).

LE K-FACTOR

Le K-Factor est un KPI permettant de mesurer la viralité d'un produit. Sa valeur représente le nombre d'utilisateurs moyen dont l'acquisition s'est faite par l'invitation d'un autre utilisateur. Concrètement, si K est égal à X, cela signifie que chaque utilisateur apporte en moyenne X utilisateurs supplémentaires au produit.

Pour calculer le K-Factor, vous devez multiplier 3 variables entre elles :

- Le pourcentage d'utilisateurs qui invitent d'autres personnes,
- Le nombre moyen de personnes invitées à chaque fois,
- Le pourcentage de personnes qui acceptent l'invitation (et deviennent donc utilisateurs).

Le rêve de tout Product Manager est, bien entendu, d'avoir un K supérieur à 1 pour que l'acquisition organique de nouveaux utilisateurs ait la forme d'une belle courbe exponentielle !

Le Net Promoter Score

Le NPS est un indicateur permettant de connaître la satisfaction des utilisateurs vis-à-vis de votre produit. Vous l'obtiendrez en demandant à vos utilisateurs de noter leur propension à recommander votre produit sur une échelle de 0 à 10 :

- De 0 à 6, vous avez les détracteurs qui peuvent vous donner du fil à retordre en terme de réputation.
- De 7 à 8, ces utilisateurs passifs sont généralement satisfaits de votre produit mais céderont facilement à la concurrence.
- De 9 à 10, vous avez vos "power users" !

Faites ensuite la différence entre les pourcentages de promoteurs et de détracteurs. Si vous obtenez un NPS positif, vous pouvez considérer que votre satisfaction utilisateurs est satisfaisante.

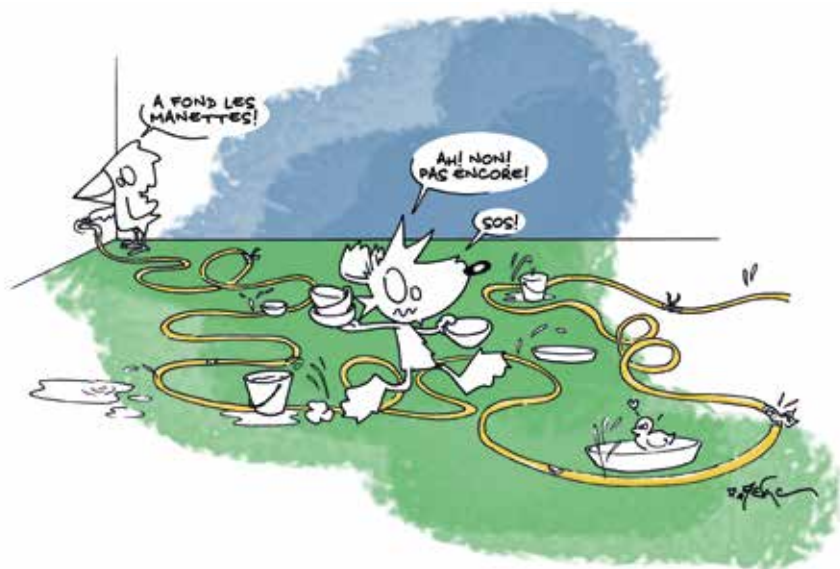
Pour favoriser la viralité, il faudra mettre en place des fonctionnalités sociales et compter sur vos "power users" pour évangéliser le marché à votre place. Attention toutefois à ne pas introduire des mécanismes viraux contre-productifs. Souvenez-vous de la myriade de jeux Facebook type Farmville vous incitant à spammer tous vos amis pour gagner de la Stamina...

La toolbox

Vous trouverez dans la règle suivante l'ensemble des outils qui peuvent vous aider dans l'implémentation globale du AARRR.

Règle n°13 :

GAGNER UN NOUVEL UTILISATEUR C'EST BIEN, LE FAIRE REVENIR C'EST MIEUX



Si votre produit nécessite un trafic récurrent des utilisateurs pour être rentable (freemium, modèle SaaS, financement par la publicité, in app purchase, etc.), vous le savez : vous pouvez acquérir tous les utilisateurs que vous voudrez, cela s'avèrera inutile s'ils ne persistent pas dans l'utilisation de votre produit. Autrement dit, vous ne connaîtrez pas de croissance sans rétention !

Le succès de votre produit passera donc par la mise en place d'un ensemble d'actions permettant d'améliorer votre rétention. Voici comment vous y prendre.

Mesurez votre rétention

Avant toute chose, vous devez définir ce que signifie la rétention pour votre produit. Certains Product Managers définissent leur rétention comme le taux de connexion à leurs applications. En réalité, cette définition est plutôt flatteuse car elle inclut tous les utilisateurs qui se sont connectés au produit sans l'avoir véritablement utilisé. C'est pourquoi, nous vous conseillons de définir un funnel de rétention, c'est-à-dire l'ensemble des actions que doit réaliser un utilisateur sur une période de temps pour être considéré comme actif (par exemple, se connecter et écouter une chanson dans le cas d'un service de musique en streaming).

Faites des cohortes !

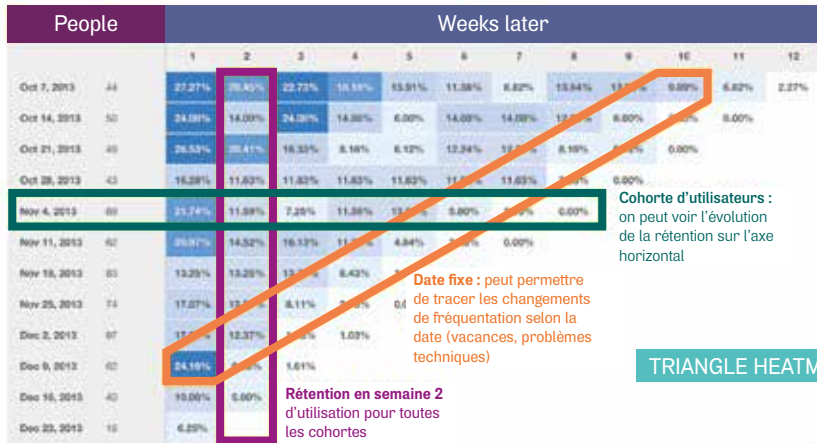
Raisonnement sur des données cumulatives (par nature en hausse !), telles que le nombre d'inscrits depuis la création de votre produit, peut flatter votre ego de Product Manager mais ne vous donnera pas d'informations sur la santé réelle de votre produit.

De la même manière, le suivi de certaines métriques telles que le nombre d'utilisateurs actifs peut vous donner l'impression que votre produit se porte bien alors qu'il n'en est rien. En effet, avoir un nombre croissant d'utilisateurs actifs est évidemment bon signe mais ne signifie pas nécessairement que vos utilisateurs continuent d'utiliser votre produit dans le temps.

Pour avoir une vision objective de l'usage de votre produit, nous vous conseillons de réaliser des analyses par cohorte. Une cohorte est un groupe d'utilisateurs partageant des caractéristiques communes sur une période donnée. Dans le jargon des produits web, le terme cohorte désigne généralement des groupes d'utilisateurs s'étant inscrits au cours du même mois ou de la même semaine.

Réaliser une analyse par cohorte hebdomadaire consiste à regrouper les utilisateurs par semaine d'inscription et comparer leur rétention au fil des semaines suivantes. Le rôle du Product Manager est de veiller à ce que chacune de ses actions concernant le produit (refonte ergonomique, ajout d'une nouvelle fonctionnalité, etc.) ou sa manière de communiquer (mise en place de plans de relance ciblés via E-Mail / SMS / notifications, etc.) améliore le taux de rétention des nouvelles cohortes.

Le meilleur moyen de comparer les cohortes entre elles est d'utiliser un "Triangle Heatmap". Il s'agit d'un tableau représentant l'évolution des taux de rétention des différentes cohortes en fonction du nombre de semaines écoulées depuis leur inscription.



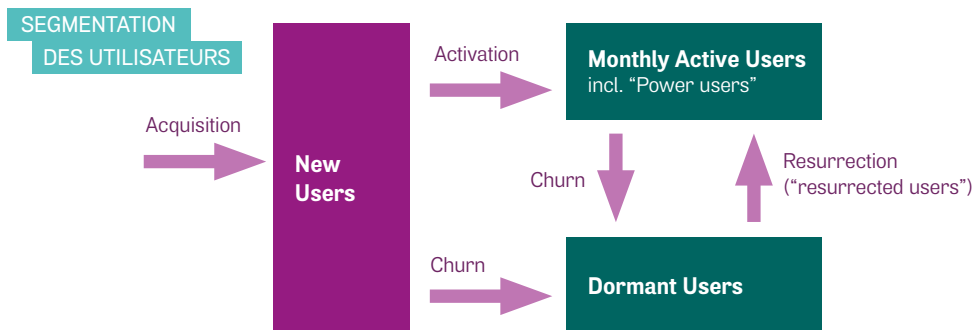
Qualifiez l'engagement de vos utilisateurs

L'analyse de la rétention passe également par la segmentation de votre base d'utilisateurs en fonction de leurs comportements au sein du produit (et pas seulement le fait qu'ils soient revenus ou non).

Une manière simple de segmenter sa base d'utilisateurs est de distinguer les trois profils suivants :

- **Power users :** ils correspondent aux utilisateurs intensifs de votre produit. Ils sont également actifs dans la communauté. Ils peuvent être early adopters pour les nouvelles fonctionnalités envisagées et faire office de pool de test.
- **Dormant users :** ils désignent les utilisateurs qui ont utilisé votre produit quelques temps mais ont fini par arrêter. Votre objectif est de comprendre la cause de leur départ et de les récupérer.

- Resurrected users : il s'agit des utilisateurs qui ont arrêté d'utiliser le service, puis qui ont été réactivés. Vous devez comprendre ce qui les a fait revenir.



Selon votre produit, les critères de segmentation de votre base d'utilisateurs peuvent être le taux de rétention, la durée moyenne des sessions, l'intensité d'utilisation de certaines fonctionnalités, le taux de transformation des relances, etc.

Améliorez votre rétention

Cela peut paraître évident, mais, avant d'essayer d'améliorer votre rétention, fixez-vous des objectifs quantitatifs. Les objectifs de rétention dépendent étroitement du secteur d'activité, du modèle économique et de la cible adressée. Nous vous recommandons d'échanger avec des Product Managers dont les produits ont des similarités avec la vôtre pour fixer une première "baseline".

Une fois vos objectifs fixés, vous pouvez jouer sur différents leviers pour améliorer votre rétention.

Alignez votre produit et votre discours marketing

Considérez la rétention comme le miroir de votre proposition de valeur. Si votre produit a un bon taux d'activation, mais un taux d'attrition élevé, c'est qu'il s'est avéré décevant : vos utilisateurs ont vraisemblablement été séduits par votre discours et le premier contact avec le produit, mais, sur la durée, ils n'ont pas trouvé les preuves de votre promesse. Si votre produit n'en est qu'à

ses débuts, vous pouvez retravailler vos discours marketing en distinguant plus clairement votre vision de vos bénéfiques produits actuels.

Relancez astucieusement vos utilisateurs

L'offre produit de notre monde numérique est pléthorique. Il est courant d'oublier un produit, alors même qu'on l'appréciait (souvenez-vous des applications de votre smartphone que vous utilisiez, il fut un temps). Vos utilisateurs vous oublient peut-être parce qu'ils sont étourdis, mais surtout parce qu'ils sont accaparés par un ou plusieurs autres produits.

La tentation de se rappeler à leur bon souvenir est alors grande : elle peut être productive, mais attention aux abus ! Une petite campagne marketing pour « ressusciter » des utilisateurs peut les inciter à se désinscrire ou désinstaller votre produit si elle est jugée trop « spammante ». Et là, vous perdez vos espoirs de réactivation.

Il faut donc essayer d'être malin et cibler quand on communique auprès d'un utilisateur ou ex-utilisateur, en qualifiant la relance :

- Nous a-t-il oublié ? Si oui, pourquoi ?
- A-t-il abandonné le service du fait d'un problème qu'il rencontrait ?

Si on fait consciencieusement ce travail, on pourra alors communiquer en essayant d'apporter de la valeur à cet utilisateur qui s'est écarté du droit chemin :

- En lui annonçant une nouvelle fonctionnalité qui pourrait l'intéresser.
- En lui précisant qu'on a réglé un problème qu'il a pu rencontrer et qu'il peut revenir.

Éliminez les goulets d'étranglement

Identifiez au sein de votre funnel de rétention, les étapes où vous perdez le plus d'utilisateurs. Vous devez comprendre pourquoi vos utilisateurs quittent votre produit prématurément. Pour cerner rapidement les améliorations possibles, menez des entretiens individuels avec certains d'entre eux (Optez pour des outils d'analytics avancés de type Mixpanel pour récupérer facilement les coordonnées de vos "dormant users").

En parallèle, cherchez à comprendre pourquoi vos "power users" aiment tant votre produit (s'ils sont friands d'une "killer feature" en particulier, cherchez à mieux la valoriser auprès des autres types d'utilisateurs).

Après avoir conduit une dizaine d'entretiens individuels, réalisez des tests A/B sur différentes cohortes et optez pour la version du produit présentant le meilleur taux de rétention.

Au-delà de l'amélioration de l'existant, vous serez peut-être amené à :

- Repenser votre expérience en profondeur (la gamification comme l'a fait Foursquare avec son système de badges est un axe possible pour améliorer la rétention).
- Ajouter des fonctionnalités "sticky" (propices à des usages intensifs) comme du "messaging" (à condition que cela soit cohérent avec votre vision produit).

La toolbox

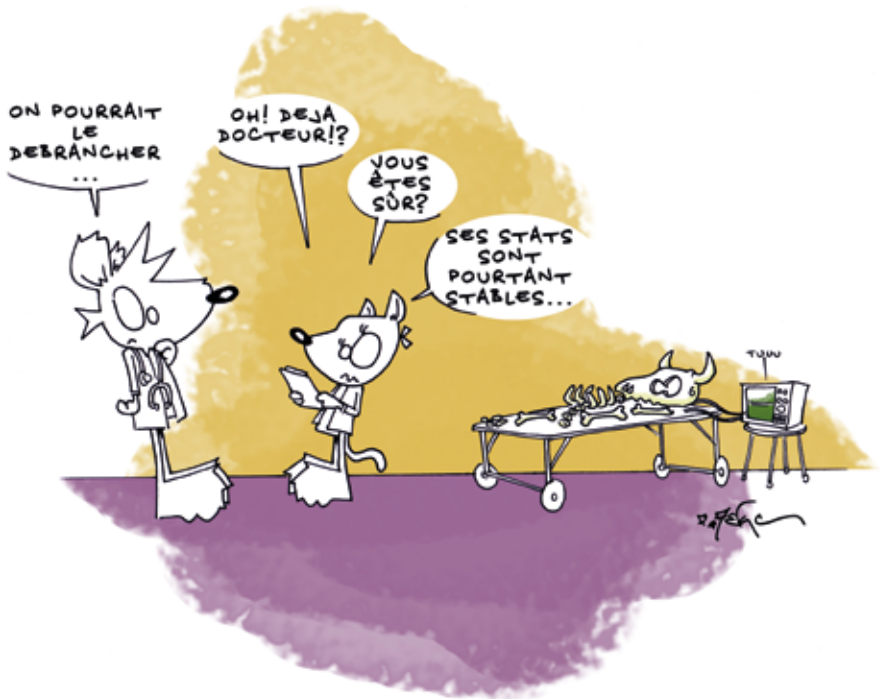
Vous devez bien sûr disposer des outils d'analytics qui vont vous permettre de suivre vos métriques d'usage. De nombreuses solutions s'offrent à vous : Google Analytics, Mixpanel, Kissmetrics, Capptain, Flurry ou encore Kontagent. La pertinence de l'outil dépendra grandement des caractéristiques de votre produit (BtoB, BtoC, monétisation, mobile ou pas, etc.), prenez le temps de benchmarker !

Au-delà de la dimension analytics pure, vous devrez aussi vous soucier des aspects suivants :

- Que disent vos utilisateurs de votre produit ? S'il s'agit d'une application distribuée sur un store, les "ratings" et les avis sont à suivre, notamment dans leur tendance générale. Pour suivre les avis sur les stores, vous pouvez utiliser des services comme AppAnnie ou Distimo.
- Si vous voulez faire en sorte que vos utilisateurs puissent vous fournir leurs retours directement, regardez des outils tels que UserVoice : une approche « helpdesk » clé en main pour un support efficace et simple à mettre en œuvre. Zendesk est aussi un autre outil qui peut vous intéresser.

Règle n°14 :

QUAND ET POURQUOI TUER SON PRODUIT ?



La plupart des Product Managers parlent de leurs produits comme du produit superbement pensé qui révolutionnera le monde. C'est normal, votre produit est votre bébé ; le fruit de plusieurs années de développement, de roadmaps longuement négociées pour en faire le meilleur produit possible. Cependant, derrière cet enthousiasme se cache une dure réalité : la plupart des produits meurent. Certains d'une mort douce, d'autres d'une fin plus brutale.

Alors que certains produits connaissent un abandon tout à fait naturel (disparition d'une technologie comme le WAP), d'autres nécessiteront que le Product Manager ou les responsables de la stratégie produit prennent les mesures nécessaires.

Si vous avez travaillé dans des startups ou dans des jeunes entreprises, vous connaissez certainement l'effet de l'accumulation de produits et de l'éparpillement de l'offre. Il s'agit d'une stratégie et d'une réaction naturelles. Ces entreprises détectent les nouveaux besoins de leur marché et créent des produits y répondant. Les produits commençant comme des PoC pour un client ou un besoin font souvent leur entrée au catalogue sans jamais en sortir. Les fonctionnalités s'accumulent alors au fil des Product Owners autour d'un concept et d'une architecture à la "quick and dirty".

Heureusement ces entreprises finissent par se structurer et ceci passe souvent par l'annonce de l'arrêt d'un ou plusieurs produits.

Pour illustrer cette situation, prenons l'exemple de 37 Signals. Le 2 février 2014, 37 Signals a annoncé à la fois l'arrêt de la quasi-totalité de ses produits à part son produit phare "Basecamp" qui représentait 87% de son chiffre d'affaires et 90% de la croissance de ses revenus, mais aussi l'abandon de son nom « emblématique » pour celui de son unique produit : à présent, 37 Signals n'est plus, il faudra dire Basecamp.

Il s'agit rarement d'une période agréable pour les Product Managers concernés : Pourquoi cette décision ? Pourquoi leur produit et pas un autre ? Comment vont-ils annoncer cela aux clients ? Que vont-ils devenir ? Et leur roadmap ? Et la fonctionnalité qui aurait pu changer la donne ?

Si même des sociétés reconnues et érigées comme exemple en passent par là, il est intéressant de mieux comprendre les mécaniques qui mènent à la mort d'un produit, afin de pouvoir maîtriser le processus et mettre en place un plan efficace pour gérer la fin de son produit.

Votre produit va-t-il si mal que ça ?

En tant que Product Manager, vous devez réfléchir à arrêter votre produit si :

- Il n'est pas viable financièrement (et ce, depuis un certain temps).
 - Le produit ne génère plus assez de revenus.
 - La marge du produit est quasi entièrement mangée par les coûts de maintenance, devenus trop élevés en raison d'une forte dette technique.
- Personne n'est là pour s'en occuper à part vous : avez-vous assez de ressources pour gérer le produit correctement, en termes d'équipe de dévelop-

pement, de qualité, de support client ? Le travail de ces équipes ne serait-il pas plus profitable si elles étaient sur un autre produit ?

- Il est en concurrence avec plusieurs produits de la même société que ce soit au niveau du marché ou des équipes.

Dans le cas de 37 signals, il s'agissait à la fois d'un manque de ressources mais aussi d'une volonté stratégique de ne pas agrandir la société. L'entreprise a préféré se concentrer sur son produit star et arrêter l'éparpillement. Dans la plupart des entreprises, il s'agit souvent d'un mélange de dette technique insurmontable, de l'éloignement de certains produits de la mission initiale de l'entreprise et de difficultés à prioriser l'allocation des équipes.

Tuer un produit : comment s'y prendre ?

Faut-il le laisser vivre en tâche de fond sans le maintenir et sans le faire évoluer tout en continuant à le vendre ? Faut-il arrêter de le proposer aux nouveaux clients mais bien s'occuper de sa base clients existante en travaillant sur des mises à jour mineures et du debug ? Faut-il complètement tuer le produit tout en laissant aux clients le temps de récupérer ou migrer leurs données ?

La décision ultime vous reviendra, mais il n'est jamais conseillé d'abandonner ses clients du jour au lendemain ou de leur donner très peu de temps pour trouver des solutions.

Quelques exemples :

- 37 Signals a fait le choix de maintenir ses autres produits pour les clients existants sans mise à jour majeure pour potentiellement les revendre ou créer des spin-off à terme, c'est le cas de Highrise depuis août 2014. L'entreprise présente l'ensemble de ses arguments aux utilisateurs et consacre une section FAQ sur son ancienne page d'accueil : <http://37signals.com/>
- Google a comme habitude de complètement tuer ses produits, en donnant assez de temps à ses utilisateurs pour se retourner et trouver des solutions à leur problème. Google a annoncé l'arrêt de Google Reader quatre mois avant

sa fermeture définitive et la suppression complète de leurs données, non sans leur présenter des alternatives : <http://alternativeto.net/software/google-reader/>
L'entreprise a fait de même pour son réseau social Orkut où les utilisateurs pouvaient télécharger l'ensemble de leurs photos et de leurs données suite à la fermeture du service cet été.

- Adobe a fait le choix de donner son framework Flex à la fondation Apache pour permettre à la communauté des développeurs de continuer le développement du produit car, en 2011, plusieurs entreprises continuaient à utiliser cette technologie pour le développement de leurs applications web. Cette dernière solution est souvent critiquée comme étant une manière de se débarrasser des produits tout en minimisant les plaintes des clients. L'Open source joue alors le rôle d'un "depot-ware" des géants du logiciel.

Tuer votre produit, n'est-ce pas couper la branche sur laquelle vous êtes assis ?

Si vous êtes Product Manager d'un portefeuille de produits, l'arrêt d'un produit vous sera plus facile. Vous aurez plus de latitude dans vos choix et dans l'allocation de votre budget global aux différents produits de votre portefeuille.

En tant que Product Manager d'un produit unique, peu de personnes oseront mettre fin à leur produit. Cela est dû à deux raisons :

- Le management par objectifs : un Product Manager vivra l'arrêt de son produit comme un échec personnel et cela pourrait avoir une incidence sur sa carrière. Cependant, dans une logique d'amélioration continue et de transparence, plusieurs entreprises décident de récompenser les décisions d'arrêt de produits à faible potentiel à condition d'argumenter sa décision et de re-diriger l'investissement initialement prévu sur un produit de remplacement, idéalement disruptif.
- L'engagement émotionnel : plus vous aurez consacré de temps à ce produit, plus il vous sera difficile de prendre la décision.

Tuer un produit est une décision difficile. Elle doit être prise de manière réfléchie en ayant bien mesuré les conséquences. Cependant, il faut savoir faire preuve de courage car c'est bien l'absence de décision franche qui peut être le plus nuisible pour vous et votre société.

Les auteurs THIGA



Hugo
Geissmann



Alexandre
Irrmann-Tézé



Nicolas
Blaisot-Balette



Audrey
Pedro



Émilie-Anne
Guerch



Benoit
Emery



Maria
Frih



Romain
Monclus



Simon
Joliveau-Breney



Cynthia
Bevilacqua



Floriane
Mignon

Les auteurs Xebia



Laurène
Vol-Monnot



Ludovic
Perot



Renaud
Chevalier



Benjamin
Moitié



Clément
Rochas



Arnaud
Brachetti

Remerciements

Nous n'aurions jamais été capable de mener cet ouvrage à son terme sans l'investissement de ceux qui l'ont relu, corrigé, illustré et mis-en-page. Nous les remercions chaleureusement !

- L'équipe marketing de Xebia, Anne Beauchart et Marina Tracco, pour leur patience infinie, leur rigueur et leurs suggestions toujours pertinentes.
- Florent Farvacque, pour ses illustrations pleines d'humour et de second degré.
- Lydie Billaud, de l'agence Michel & Michel, pour avoir travaillé d'arrache-pied jusqu'à la dernière minute et pour avoir supporté nos changements d'avis incessants (à ce stade là, ce n'est même plus de l'agilité).
- Les coachs Agiles de Xebia, pour leur relecture attentive et leurs conseils avisés.
- Luc Legardeur, président de Xebia, pour le dynamisme et l'énergie qu'il a toujours su insuffler à ceux qui l'entourent.

