Chapitre 4
Stratégie pour un logiciel SaaS durable

1. Construire et piloter sa roadmap

1.1 Les enjeux critiques

Comme nous l'avons vu plus haut, la réussite dans le SaaS repose tout d'abord sur une croissance rapide, donc sur la vitesse d'exécution initiale puis sur la capacité à rendre cette croissance durable.

Pour cela, un certain nombre de critères doivent être bien identifiés, portés au plus haut dans les choix d'architectures et pilotés dans la roadmap qui doit devenir la boussole de l'ensemble de l'entreprise.

Les enjeux critiques qui doivent être pilotés dans la roadmap tournent autour de l'intégrabilité, l'ouverture du système, la sécurité, la performance, la User Experience, la maîtrise de la dette technique et selon le secteur métier, on peut retrouver des enjeux réglementaires aussi.

Il est donc essentiel que la construction et le pilotage de cette roadmap soient un processus connu de tous et collaboratif.

1.2 Construire sa roadmap

1.2.1 Les trois niveaux de la roadmap

La roadmap doit porter la vision stratégique de l'entreprise et sa déclinaison en exécution opérationnelle par raffinages successifs.

Il est important de considérer qu'une roadmap n'est pas un objet figé dans le marbre : au contraire, elle évolue en permanence et c'est plutôt un signe de bonne santé, à condition de ne pas tomber dans l'excès inverse révélant un manque de constance dans la stratégie globale produit.

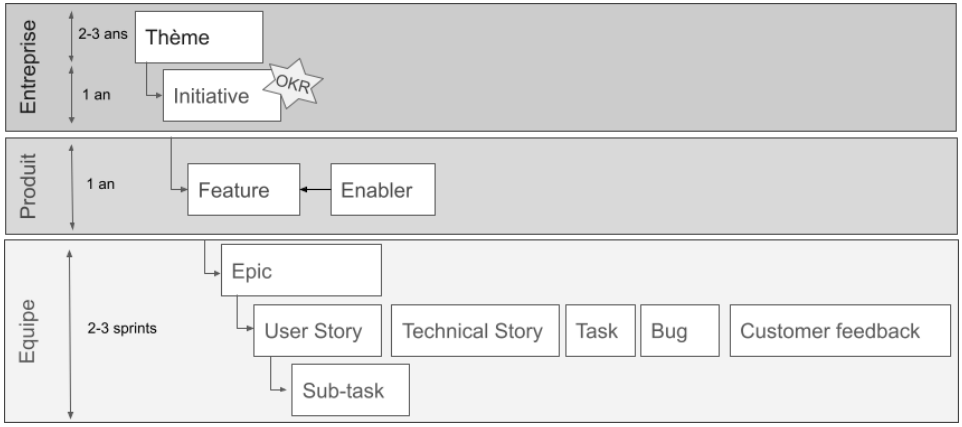
Voici les trois niveaux que nous proposons de mettre en œuvre dans une roadmap :

- Tout en haut se trouve le niveau Entreprise pour lequel nous proposons une arborescence à deux niveaux :
  - Thème : on décrit ici les éléments du Business Plan de l'entreprise qui sont les objectifs stratégiques à deux ou trois ans en général. Ce niveau est porté par la direction marketing et peut aussi être dénommé Objectif (ou Goal) dans certaines organisations.
  - Initiative : une initiative est à considérer comme un ensemble d'améliorations consistantes entre elles dans un thème donné permettant de délivrer une valeur métier. On décline les initiatives en général sur une période annuelle et on les rattache systématiquement à un thème de l'entreprise. Un budget est généralement alloué à une initiative et un retour sur investissement est décrit. Le pool d'initiatives n'est pas une liste figée, il évolue en permanence au gré du raffinage de la stratégie marketing. Le niveau initiative ne préjuge pas d'une déclinaison au niveau implémentation produit, il doit rester centré sur un besoin et identifier les contraintes haut niveau (ex. contraintes de date liées à un événement réglementaire, ou volumes de données ou de clients…). Une initiative se matérialise en général sous forme de fiche qui ne dépasse pas une page A4, et qui décrit autant un scope qu'un hors-scope (qui peut avoir été déterminé par un Story Mapping). On peut y associer un OKR (*Objective Key Result*), méthodologie popularisée par Google, de manière à alimenter les revues des objectifs de l'année en cours. Le niveau initiative est porté par le Product Management.

- Niveau Produit :
  - Feature : une fiche Feature décrit une nouvelle fonctionnalité associée à un produit et répondant à l'une des initiatives du Product Management. Chaque nouvelle Feature est systématiquement rattachée à une initiative de manière à éviter les Features orphelines et donc potentiellement non alignées avec les priorités de l'entreprise. La fiche Feature doit contenir les éléments qui vont servir d'entrants à la R&D pour alimenter le Product Owner et concevoir l'architecture qui supportera la Feature. Par exemple, le fait d'indiquer qu'un service backend doit supporter jusqu'à un million de requêtes par jour est un élément structurant de l'architecture à concevoir. La fiche Feature doit décrire les dépendances fonctionnelles et techniques avec d'autres produits ou systèmes : par exemple, dans le cas de l'appel d'un système externe par une API. Le niveau Feature est sous la responsabilité du Product Owner.
  - Enabler (optionnel) : un enabler est une Feature à visée plutôt technique, caractérisée par le fait qu'elle est nécessaire à débloquer une ou plusieurs autres Features. Ces Enablers peuvent être des éléments d'architecture centraux, comme l'instauration d'un nouveau middleware. Dans une approche DevOps, c'est typiquement dans cette enveloppe que l'on va identifier les travaux autour de l'usine logicielle. Le fait de les mettre à ce niveau permet de les rattacher à une initiative ayant une portée métier en évitant de créer des initiatives marquées comme techniques : l'utilisation d'initiatives techniques peut en effet s'avérer un piège dans des domaines très technologiques où on pourrait confondre l'apport technologique avec la cible fonctionnelle. Le caractère optionnel vient du fait que c'est en général dans une équipe à l'échelle que l'on va rencontrer ce besoin, et ce niveau permettra d'identifier de façon explicite la part de travaux directement liée à une Feature fonctionnelle des autres. Ce niveau est sous la responsabilité d'un Leader Technique ou d'un Architecte.
- Niveau Equipe, sous la responsabilité du Product Owner :
  - Epic (optionnel) : l'Epic permet un regroupement logique de User Stories qui adressent une même fonctionnalité. On la met en œuvre en cas de fonctionnalité de grande taille, ou pour des raisons de gestion de dépendance optimisée.

- User Story : elle décrit une fonctionnalité autonome ayant une valeur utilisateur propre. Elle est théoriquement directement livrable unitairement. L'ensemble des User Stories est la base de la constitution de la release note qui sera affichée au client. Des critères d'acceptation testables sont décrits pour chaque User Story.
- Technical Story : elle est utilisée pour du développement qui contribue à la fonctionnalité mais qui n'a pas de valeur directement utilisable par l'utilisateur final. Elle n'apparaît pas dans les release notes.
- Task : la Tâche consiste à identifier toute activité de l'équipe qui ne soit pas directement du code, comme réaliser une journée de tests, une étude bibliographique, un atelier avec le client ou une journée de formation.
- Bug : tout élément de non-qualité détecté en interne, automatiquement par la CI ou manuellement par un équipier. Le bug ne doit pas être utilisé pour passer artificiellement une Story à Done et garder "sous le coude" le bug à corriger plus tard. Le cas d'usage classique est le test KO en CI ou une découverte d'une anomalie par qui que ce soit dans l'organisation. Un bug ne porte pas de valeur estimée en story point, car il n'a pas vocation à donner de la valeur à la non-qualité d'une Feature.
- Customer Feedback : il s'agit d'une anomalie ou d'un retour client ayant été rebasculé en Niveau 3 depuis le portail support. Le fait de le distinguer du type Bug n'est pas obligatoire mais peut permettre d'instaurer des workflows dédiés à chacun. En effet, la criticité d'un bug client peut nécessiter une priorisation immédiate et des éléments de notifications dédiés en lien avec le workflow du support client. Il y a débat sur le fait de donner une estimation de valeur à un Customer Feedback. En effet, un bug client n'est pas seulement un "crash" mais relève parfois de la mise au point fonctionnelle voire d'une reconception partielle de la fonctionnalité. Dans ce cadre, on peut considérer qu'on est dans un processus agile de livraison continue et d'adaptation selon les feedbacks.

– Sub-Task : il s'agit de la maille Agile de découpage des tâches nécessaires pour réaliser une User Story. Classiquement, on trouve ici les tâches de type backend, frontend, création des tests autos, documentation, etc. L'idée étant de mettre en visibilité le travail collaboratif de l'équipe pour sécuriser au plus tôt les User Stories critiques du sprint en cours. Certaines équipes d'ailleurs réalisent leur Daily Meeting et suivent leur Kanban sur les sous-tâches. Le découpage en sous-tâches est souvent un signe très positif de capacité de collaboration et de maturité technique et fonctionnelle d'une équipe Agile : il se réalise de préférence lors de la séance de Sprint Planning.



Il est important de considérer que les trois niveaux de construction de la roadmap (Entreprise, Produit et Équipe) fonctionnent par raffinages successifs de manière synchrone, en mode "juste à temps" comme le préconise la philosophie Lean, c'est-à-dire ni trop tôt ni trop tard.

En pratique, on cherchera à avoir un backlog de niveau entreprise raffiné à deux ou trois années, un backlog de niveau produit raffiné à une année et un backlog de niveau équipe avec deux à trois sprints d'avance.

Le fait de construire une arborescence verticale de ce type porte un enjeu d'alignement de tous les intervenants vis-à-vis de la raison d'être de l'entreprise. Tout développeur peut ainsi à partir de la User Story sur laquelle il travaille, en déduire la Feature et l'initiative, et de fait le thème de la roadmap à laquelle il est en train de contribuer. Dans le sens inverse, un Product Manager peut apprendre de la déclinaison opérationnelle des initiatives qu'il a priorisée et visualiser la portée de chacune en termes de charge de travail.

1.2.2 Les métadonnées pour les enjeux transverses

L'arborescence verticale vue plus haut a pour but de décrire une stratégie produit et doit être complétée à chaque étage par des métadonnées qui vont ainsi permettre de mettre en œuvre des vues horizontales. En effet, la roadmap a pour objectif de visualiser de manière explicite et aisée les items associés aux enjeux critiques décrits dans le chapitre Une usine logicielle SaaS. Cela permet ainsi aux directions transverses de monitorer les travaux sans avoir à créer et gérer leur propre roadmap parallèle. Par exemple, le RSSI doit pouvoir facilement visualiser l'avancement des items qui sont associés au tag "Sécurité" et à ses dépendances. Il est donc important que l'entreprise identifie correctement un jeu de tags ou labels à faire porter à chaque élément de la roadmap pour offrir cette possibilité de vision roadmap horizontale.

Faire en sorte de centraliser totalement la roadmap et éviter les doublons ou triplons (ou plus encore) est un enjeu d'alignement qui doit être surveillé de près par la direction produit.

1.2.3 MVP et Story Mapping

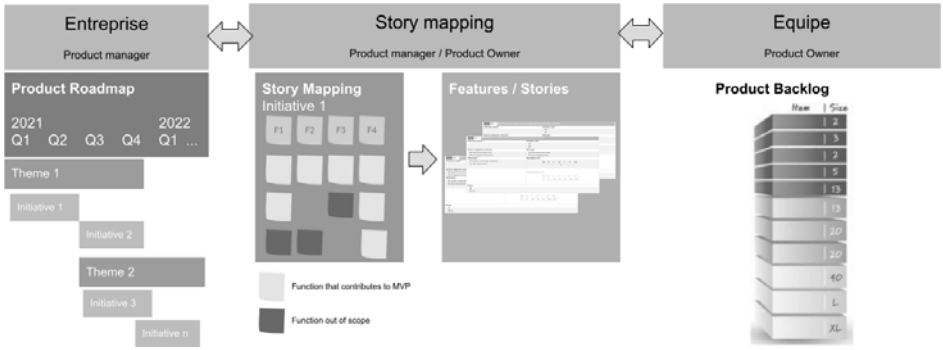
Aligner les parties prenantes de haut en bas et de bas en haut exige notamment de bien définir derrière les titres la portée de chaque item, quel qu'en soit son étage dans la roadmap.

La technique de Story Mapping est en ce sens un outil collaboratif et puissant pour déterminer ce scope et le découpage des releases à venir.

Les principaux objectifs d'un Story Mapping sont les suivants :

– Identifier les principales fonctionnalités, leurs relations entre elles et leur décomposition.

– S'assurer de la complétude du besoin fonctionnel.
– Aider à réaliser les priorisations haut niveau.
– Assurer une consistance du plan de release.



L'atelier de Story Mapping va se dérouler en plusieurs grandes étapes qui vont être travaillées dans cet ordre, pour une durée d'environ 2 h :

– Étape 1 : donner un titre au Story Mapping. Cette étape exige souvent 15 minutes pour aligner les participants sur le périmètre du Story Mapping.
– Étape 2 : identifier les prérequis et conditions de départ du processus.
– Étape 3 : décrire les conditions de sortie du processus.
– Étape 4 : identifier les personas intervenant directement dans le processus à analyser.
– Étape 5 : lister les activités qui vont s'enchaîner dans un ordre chronologique.
– Étape 6 : décomposer ces activités en Stories à classer de la plus importante à la moins importante de haut en bas.

À l'issue du Story Mapping, le squelette de la roadmap et du backlog est ainsi constitué. Les Stories indispensables constitueront le backlog MVP de la Feature, les Stories souhaitables seront mises dans une deuxième Feature à planifier pour une prochaine release, les Stories optionnelles seront mises en attente dans le backlog.