

Agile Scrum (approfondissement)

Thomas Aldaitz
taldaitz@dawan.fr

Plus d'informations sur <http://www.dawan.fr>

Contactez notre service commercial au **09.72.37.73.73** (appel gratuit depuis un poste fixe)



Objectifs

- Maîtriser les concepts de gestion de projets agiles

Plan

- Gestion d'un projet agile Scrum
- Besoins et planification
- De l'analyse à l'implémentation
- Outils de gestion

Rappel du cadre Scrum



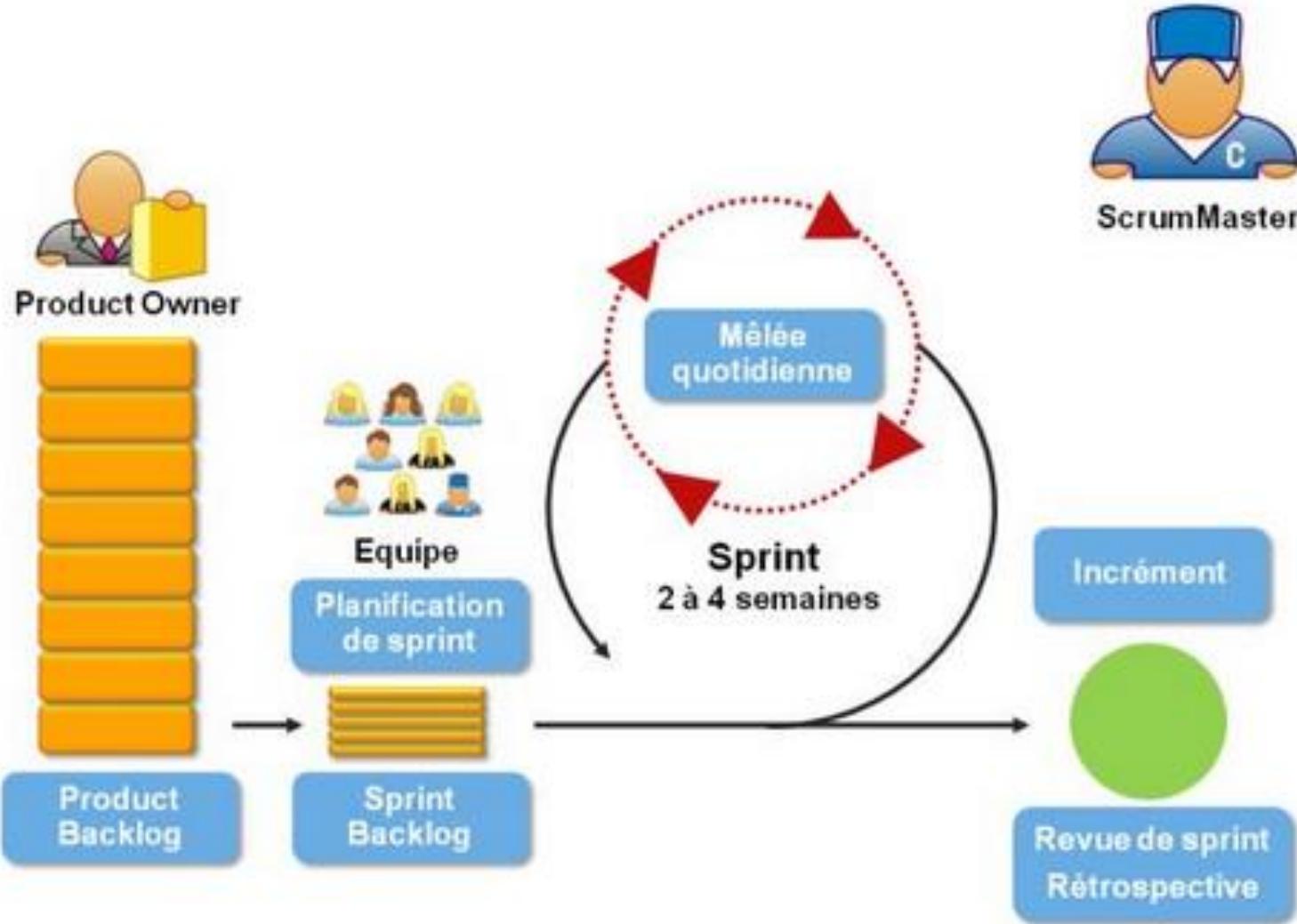
Scrum

- Cadre ou « framework » de gestion de projets :

<https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-French.pdf>

- Développé en 1993 par Ken SCHWABER et par Jeff SUTHERLAND
- Méthode itérative (sprints) et incrémentale
- Équipes de 3-9 personnes
- Mécanismes d'extension possibles (Scrum of Scrums).
- Méthode agile la plus utilisée avec eXtreme Programming

Scrum - Cycle de vie



Scrum - Phases

• Phase Initiale :

- Planning.
- Mise en place d'un backlog (liste des tâches à effectuer).
- Définition de l'équipe.
- Analyse des risques - Budget.

• Sprint :

- 2 à 4 semaines isolées de toute influence extérieure.
- Un sprint backlog est suivi et réalisé.
- Réunion quotidienne Scrum.
- Réunion post-sprint de présentation des résultats.

• Clôture :

- Documentation finale.
- Livraison

Scrum

Réalisation d'un Sprint



• Activités & organisation de l'équipe

- Elle s'auto-organise, estime, fait les choix et se répartit les tâches pour délivrer le produit partiel
- Elle doit être dans un environnement optimisé pour atteindre l'objectif
- Les tableaux de bord et indicateurs sont actualisés quotidiennement par le Scrum Master et permettent de suivre l'avancement de l'itération.

• Méthodes

- L'estimation, l'analyse et la conception sont réalisées avec des méthodes agiles (cartes, index, croquis UML, ...)
- L'accent est mis sur les tests (Test Driven Developpement > mesure d'avancement par les tests, issu de l'XP)

Scrum

Estimations, planification, avancement

- Estimations :

- par analogie de préférence (utilisation du réalisé comme référence), ou par point de fonction
- Utilisation de méthodes agiles ("planning poker")

- Planification

- Le planning doit permettre à l'équipe de "souffler" entre 2 sprints
- Exemple : démos & rétrospective le vendredi matin pour le sprint n, planification du sprint n+1 le lundi matin ▶ vendredi AM libre pour la veille, ...

- Les estimations et la planification se basent sur le calcul de la vélocité = (nb de jour/homme disponibles) x (coefficients d'effort)

- Le coefficient d'effort est une estimation sur la concentration de l'équipe. Il peut être évalué à partir des sprint précédents.

- Par exemple :



© SCRUM and XP from the Trenches, Henrik Kniberg

Gestion de projets agiles



Comment recueillir et formaliser le besoin ?



Recueil des besoins

Problème : attachement à une phase amont durant laquelle on souhaite recueillir et figer l'exhaustivité des besoins.

Recueillir efficacement les besoins, c'est avant tout améliorer la qualité des échanges entre toutes les parties prenantes.

Difficultés

- Mauvaise communication :
 - - acheteurs / utilisateurs
 - ambiguïté de formulation = différentes interprétations
- L'illusoire exhaustivité :
 - - vouloir décrire intégralement les besoins dès le début.
- Défaillance du client :
 - - expression des besoins réduite au strict minimum.
 - - indisponibilité du client

Démarche de recueil

- Partager une vision
- Collaborer avec le client
(compétences techniques / comportementales)
- Faire émerger le besoin
- La boucle de feedback

Partager une vision

- Vision du produit / projet = orientation générale donnée à l'équipe (objectif à atteindre)
- La vision repose sur une étude d'opportunité et/ou analyse des besoins.
- Cette vision doit être communiquée par le client et partagée par tous (l'absence de vision peut être un facteur d'échec).

Partager une vision Forme



- **Pourquoi ?** Quel est l'objectif stratégique du client ?
- **Quoi ?** Quels sont le périmètre et les contraintes actuels et futurs, du projet ?
- **Qui ou pour qui ?** Quelles sont les parties prenantes du projet ? Qui va utiliser le produit ? A qui va bénéficier le projet ? A toute personne impactée ou qui peut impacter le déploiement d'un produit, y compris l'équipe de réalisation.
- **Comment ?** Comment l'équipe va-t-elle réaliser le produit : va-t-elle le faire ? Le faire-faire ? Ou l'acheter ? Il s'agit ici de proposer une solution technique, un calendrier, un budget, de recenser les ressources nécessaires, de déterminer la méthodologie qui sera mise en œuvre.

Partager une vision

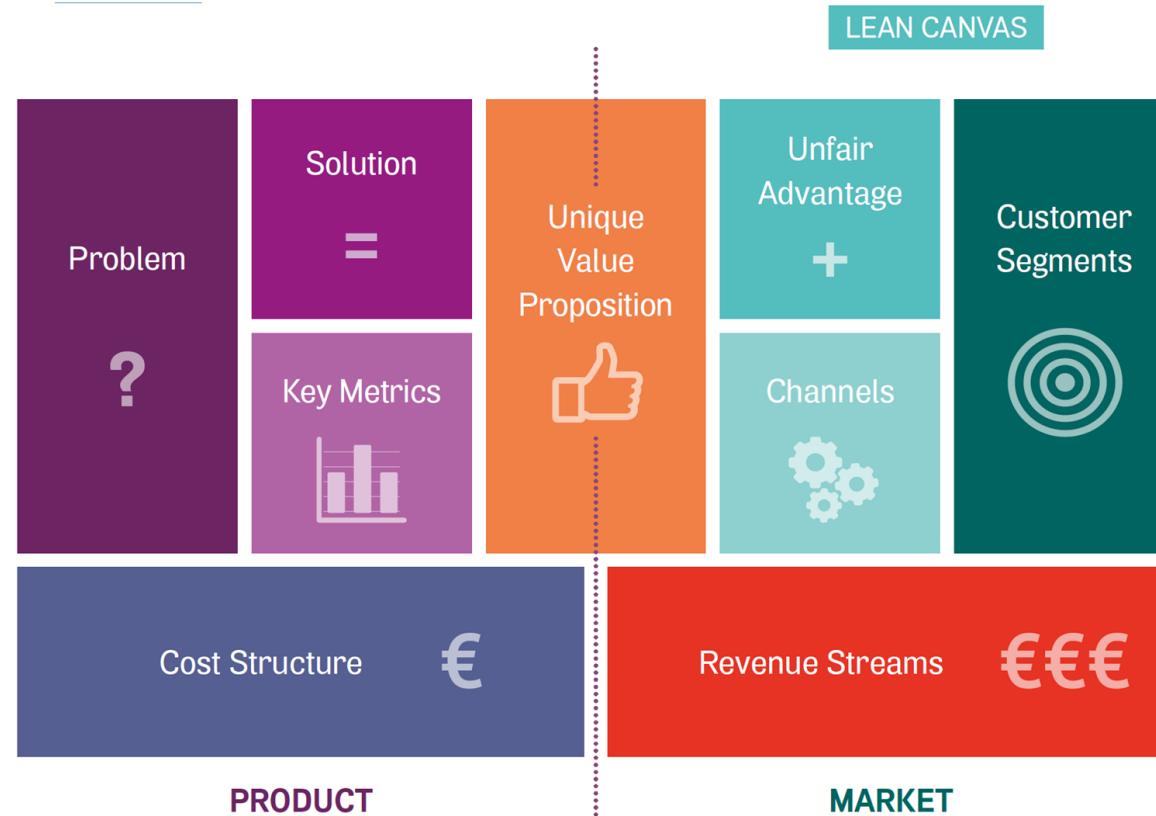
Forme (2)

Exercice de l'Elevator statement (Geoffrey Moore)

- L'utilisateur ...
- ... a un besoin ...
- ... que le produit que nous allons développer, qui est un produit de type ...
- ... va satisfaire ainsi ...
- ... contrairement à d'autres, ou par rapport à la solution existante ...
- ... donc se positionne de cette façon

Partager une vision Méthodologie (Lean Canvas)

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



Partager une vision Méthodologie (Lean Canvas)

- **Segment utilisateur** : qui sera la cible de notre produit ? Qui sont nos potentiels early adopters ?
- **Problème** : quel est le top 3 des problèmes que nous cherchons à résoudre pour nous, early adopters supposés.
- **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 ?
- **Solution** : quelles sont les 3 grandes fonctionnalités qui vont répondre aux problèmes de nos early adopters ?
- **Canal** : quels sont les canaux gratuits et payants que vous pouvez utiliser pour atteindre vos futurs utilisateurs ?
- **Métriques** : quelles sont les métriques que vous allez mesurer pour valider ou invalider vos hypothèses ? (NPS, AARRR, chiffre d'affaires...)
- **Coûts** : quels sont vos coûts fixes et variables ?
- **Revenu** : quel est le modèle économique ?
- **Avantages concurrentiels** : qu'est ce qui vous rend plus performant que vos concurrents pour traiter les problèmes identifiés ? (Connaissance technique, carnet d'adresse, marque).

Partager une vision Toolbox



- Product Vision board :

<http://www.romanpichler.com/tools/vision-board/>

- Business Model Generation :

<https://strategyzer.com/canvas/business-model-canvas>

- Outils en ligne : <https://next.canvanizer.com/>

Collaborer avec le client

- Client = un représentant des utilisateurs du produit.
- Plusieurs niveaux de représentation selon la structure ou la taille de l'organisation : chef de projet, utilisateur, utilisateur final, commanditaire, maître d'ouvrage, responsable d'applications, product owner, ...
- Pour une collaboration fructueuse, le client doit développer 2 types de compétences :
 - - techniques : expression et recueil des besoins.
 - - comportementales : favorisant la collaboration.

Collaborer avec le client

Compétences techniques du client

- **Il communique sa vision** : une échéance à respecter comme seul objectif n'a pas beaucoup de sens si l'équipe ne connaît pas les enjeux au delà de l'échéance.
- **Il décrit les fonctionnalités** : fournir une description apportant de la valeur ajoutée aux utilisateurs. Une description suffisamment détaillée pour une bonne compréhension mais conserve une granularité grossière pour pouvoir être estimée en charge.
- **Il connaît les priorités** : les éléments ayant le plus de valeur ajoutée (tout ne peut être prioritaire). Il est capable d'arbitrer, en tenant compte des contraintes techniques (risques, impacts, coût de développement..).
- **Il peut changer d'avis** : hiérarchiser les activités est une tâche difficile. Le client peut changer d'avis en cours de projet et changer le périmètre fonctionnel (avec des arbitrages).

Collaborer avec le client

Compétences comportementales

ou savoir-être



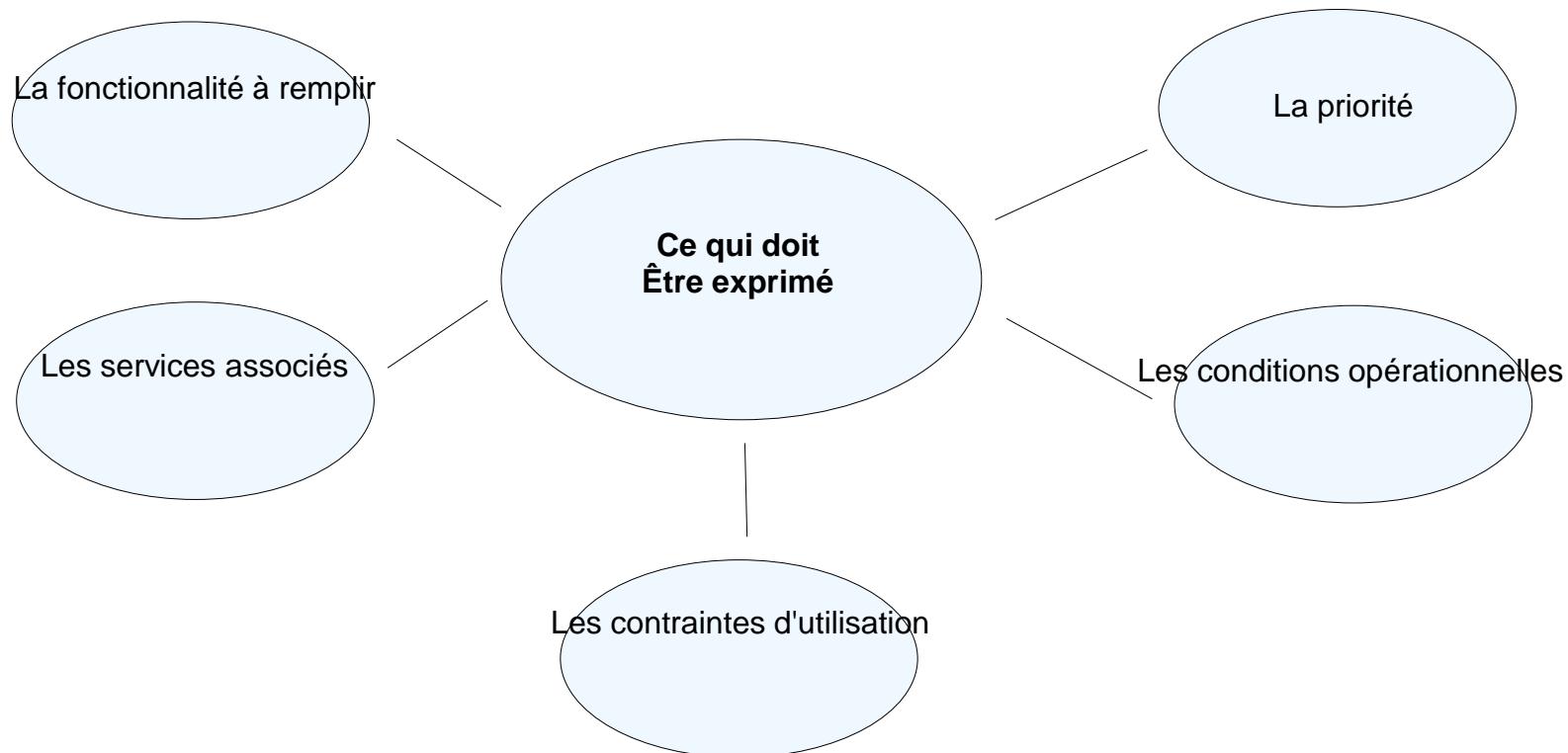
- **Il collabore** : fortement impliqué, participe aux réunions, partie prenante de l'équipe grâce à son expertise métier.
- **Il est exigeant** avec l'équipe de réalisation mais exprime sa confiance en l'équipe (acquise grâce à des livraisons fréquentes). Il doit voir des résultats intermédiaires concrets et tester régulièrement.
- **Il soutien l'équipe et a de la reconnaissance.**

Sans compétence spécifique, en adoptant un comportement collaboratif, le client obtient de meilleur résultat de l'équipe (partenariat fonctionnel / technique).

Faire émerger le besoin Ce qui doit être exprimé

- Un besoin est défini à partir des objectifs et de la vision.
- Un besoin n'est pas uniquement une fonction assurée par le système (enregistrer une commande, créer une fiche produit) ; c'est aussi la capacité du système d'assurer efficacement cette fonction, les conditions opérationnelles (disponibilité, évolutivité, performances) ; le besoin se définit également par les services associés (modalités d'exploitation, support utilisateurs) et par les contraintes d'utilisation (ergonomie, organisation des utilisateurs, leur implantation géographique,...)

Faire émerger le besoin Ce qui doit être exprimé (2)



Techniques de recueil

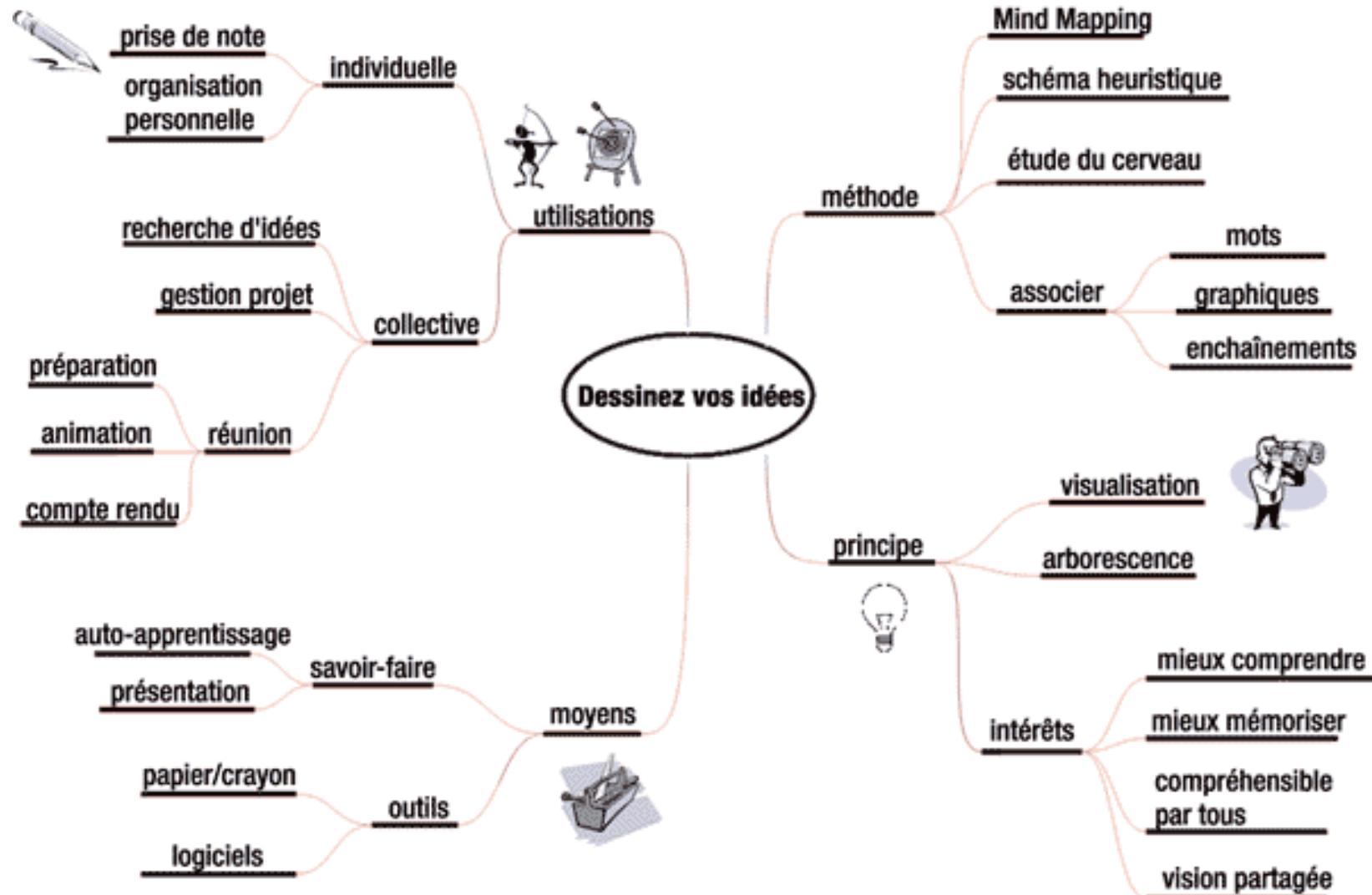
- Plusieurs techniques sont possibles (combinées ou utilisées individuellement) :
 - **En amont du projet** : brainstorming, analyse de la concurrence (benchmark).
 - **Pour affiner le besoin** : interview, atelier facilité (workshop).
 - **Pour la description détaillée** : analyse de l'existant, observation du comportement de l'utilisateur (en situation), qcm.
- Le choix tient compte :
 - des utilisateurs (nombre, localisation, disponibilité),
 - de l'étape de recueil (largeur, profondeur),
 - des caractéristiques du produit (interface, traitements internes)

Techniques de recueil

Brainstorming - Benchmark

- Brainstorming : réunions courtes où tout les participants s'expriment + un facilitateur.
 - Idéal pour défricher les besoins encore flous ou mal organisés.
 - Outils : Mind Mapping (Mind Jet, X-Mind, Coggle etc...)
 - Représenter graphiquement des idées, concepts ou informations => partir d'une idée centrale ou une question et organiser/représenter l'information de manière visuelle et structurée.
- Benchmark : Analyse de la concurrence

Techniques de recueil Brainstorming (Mind Map)



Techniques de recueil Interview



- Technique directe pour approfondir le besoin. Préparer au préalable un questionnaire pour guider et optimiser l'entretien.
- Penser à croiser les données de plusieurs entretiens
- Technique des 9 cases (Solution Selling)

Techniques de recueil Interview (Techniques des 9 cases)

	Quel est le problème ?	Qui est impacté ?	Visualiser la solution
Questions ouvertes : « Dites-moi... » « Racontez-moi... » « Et puis...? » => HISTOIRES	1	4	7
Contrôle : « Combien..? » « Quand..? » « Où...? » => FAITS	2	5	8
Validation : « Si je comprends bien... » => Si non : retour aux questions ouvertes. => Si oui : passer à la question suivante.	3	6	9

Techniques de recueil

Workshop



- Forum d'échanges
- Groupe de travail réuni sur un thème donné (architecture ou autre).
- Un animateur organise les processus et prévoit l'équipement adéquat

Techniques de recueil

Analyse de l'existant

- Examiner les applications existantes pour évaluer les forces et faiblesses.
- Mettre au point la stratégie d'évolution (gap analysis) :
 - Quelles sont les fonctions à réutiliser ?
 - Que doit-on remplacer ?
 - Comment les améliorer ?
 - ...

Techniques de recueil Observation du comportement



- Observer l'utilisateur en situation sur son poste de travail : pratiques, outils complémentaires, ergonomie
- Possibilité d'émettre un questionnaire à un ensemble d'utilisateurs avec des questions ouvertes/fermées.
- Dans une approche agile, on privilégie le contact direct.

Techniques de recueil

Organiser les rôles

- Associer un besoin à un contributeur.
- Formalisation : **Matrice RACI**
- R : responsable de la description du besoin
- A : approbateur, de qui R doit obtenir la validation
- C : contributeur, à qui R peut demander une contribution
- I : Informé de l'existence d'un besoin
- Outils : Tableau Excel (modèle RACI existant)

Techniques de recueil

Organiser les rôles - RACI

	Direction	Responsable Produit	Responsable Informatique	Responsable Projet	Développeur	Utilisateur Clé	Équipe Application	Formateur
Analyse	A	R	C	-				
Définition du projet	C	A,R	I	C			I	
Business Plan détaillé	A	C	R	I			C	
Test plan	A		C		R		C	
Création de la plateforme	A		I	I			R	
Développement du programme	A			R R				
Définition des tests	A			I I	R		I	
Réalisation des tests	A				R			
Migration des données	A		I	I I		R		
Formation des Utilisateurs	A						R	
Décision de lancement (ou non) de l'application	A	R	C	-	-	-	I I	
Mise en Production	A	I	-	-	-	R	I	

Construction de la 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 œuvre pour les atteindre, ainsi que les risques identifiés.
Ne pas confondre la roadmap pas avec une liste sans fin de fonctionnalités prévues pour votre produit.

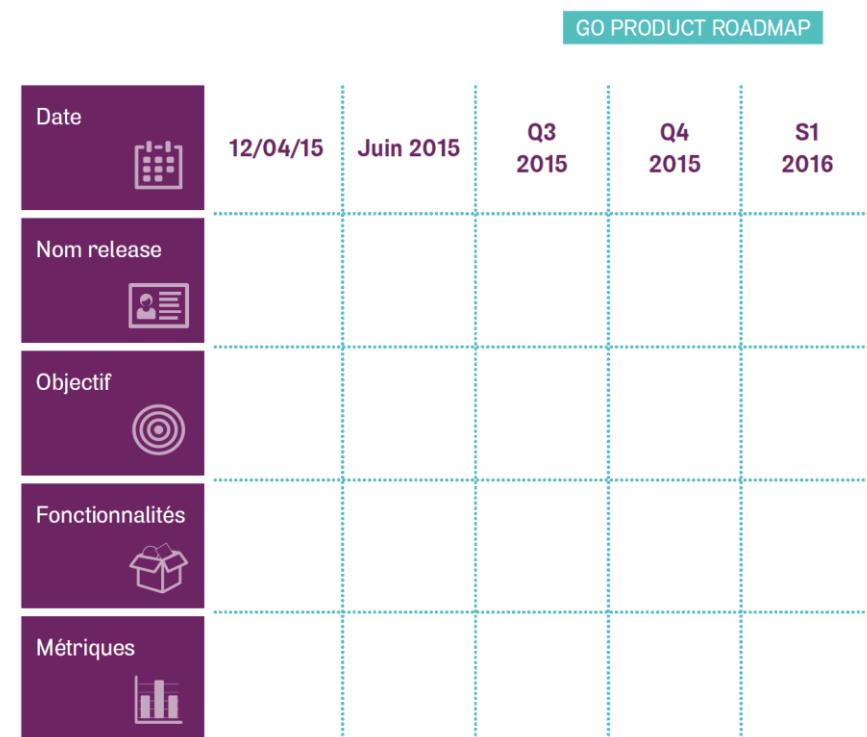
Construction de la Roadmap

Forme



- 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 de la livraison,
- les fonctionnalités incluses,
- les métriques à suivre.



- 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.

Construction de la Roadmap Outils



- 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).
- 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) découplant le parcours utilisateur en étapes chronologiques et un axe (vertical) détaillant ces étapes en fonctionnalités.
- L'utilisation conjointe de ces deux outils vous permettra de construire une roadmap pertinente.

Impact mapping

Objectifs => Fonctionnalités

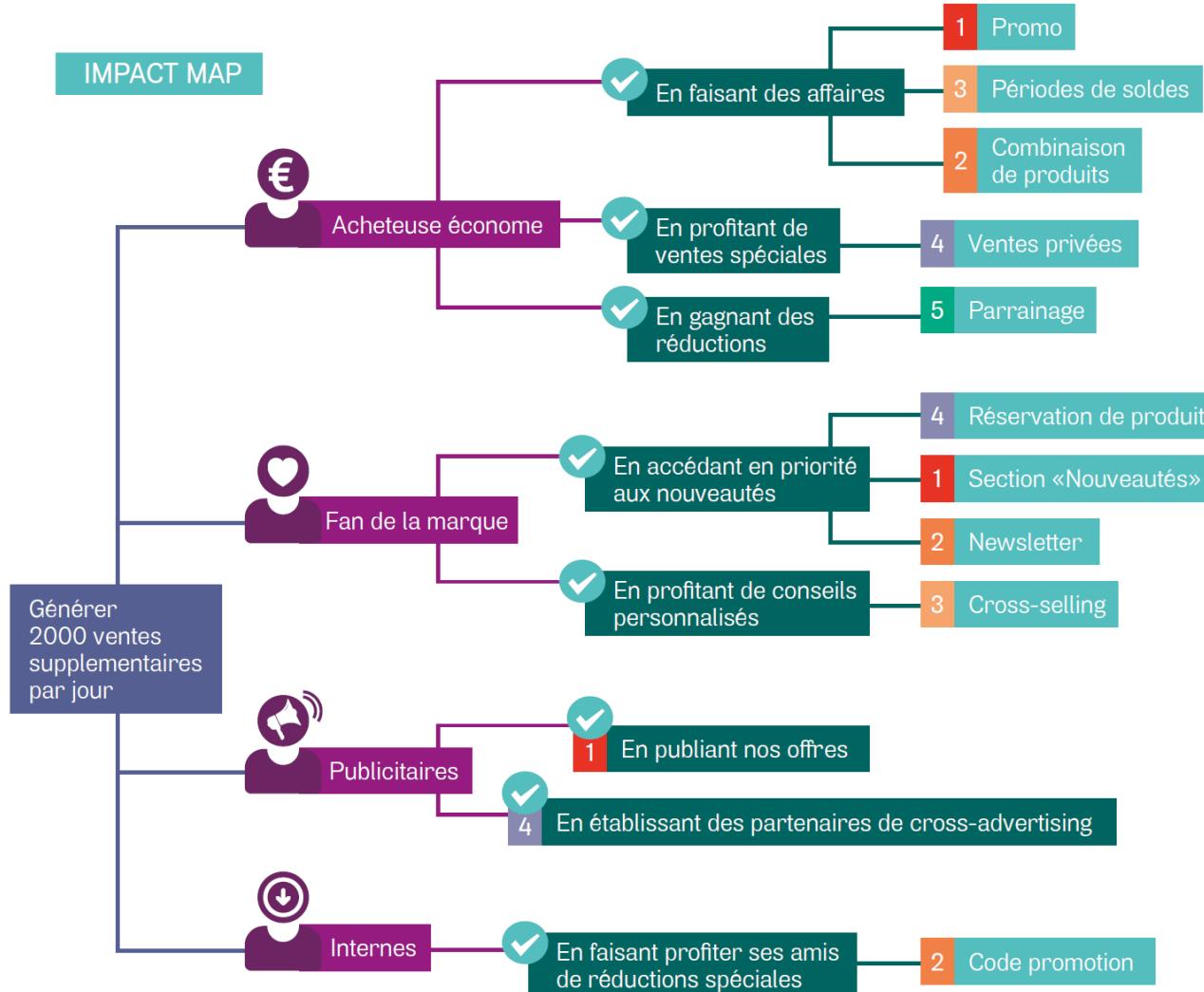


Permet de faire découler les fonctionnalités des objectifs, et non le contraire. La création d'une impact map déroule les sujets de la façon suivante :

- **Les objectifs** : Pourquoi souhaitez-vous créer ou faire évoluer votre produit ?
- **Les acteurs** : De qui avez-vous besoin pour atteindre vos objectifs ? Quelle est votre audience ?
- **Les impacts** : Comment souhaitez-vous modifier le comportement des acteurs afin d'atteindre votre objectif ?
- **Les fonctionnalités** : Quelles fonctionnalités ou actions devez-vous mettre en place pour obtenir les comportements identifiés ? La granularité est ici celle du parcours utilisateur.
- **Toolbox** : XMind, Mindmup

Impact mapping

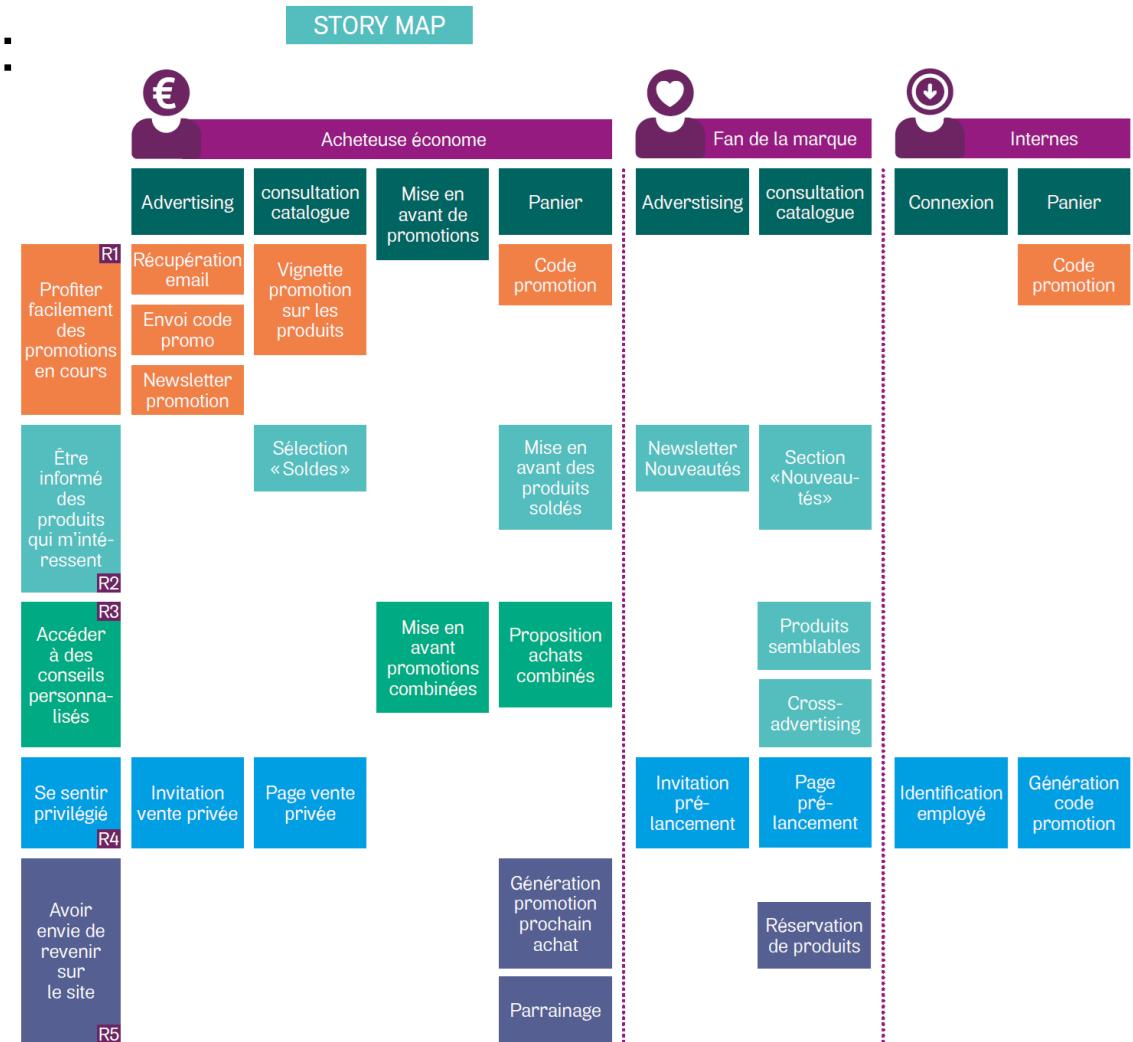
Exemple : site e-commerce



Story mapping

- Identifier et de structurer les fonctionnalités suivant 2 axes : un axe horizontal découplant le parcours utilisateur en étapes chronologiques et un axe vertical détaillant ces étapes en fonctionnalités.

- Regrouper les fonctionnalités en **MMFs** (Minimum Marketable Features) puis ordonner ces dernières pour définir des releases.



Story map => Roadmap

- Extraire la 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.
- Il est impossible de prévoir une date de livraison précise tant que le Product Owner n'a pas une idée de la vitesse 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.
- **La roadmap n'est pas gravée dans le marbre, bien au contraire. Elle a vocation à évoluer avec votre produit.**

Story map => Roadmap (2)

GO PRODUCT ROADMAP					
Date	12/04/15	Juin 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-privee ▶ 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>

Formaliser les besoins

- On formalise le besoin pour :
 - - ne pas avoir de déperdition de données
 - - avoir une traçabilité
- Dans une approche agile :
 - - La traçabilité est simplifiée (itérations courtes).
 - - La formalisation est minimale et éphémère. Cette dernière n'est pas nécessairement conservée une fois le besoin traité et satisfait.
- Plusieurs approches : IEEE, Use Cases, User Stories.

Formaliser le besoin

Approche IEEE



- Norme IEEE830-1998 : Pratiques recommandées pour la spécification des exigences logicielles (SEL).
- Définition d'une exigence (source IEEE) :
 - (A) Une condition ou capacité requise par l'utilisateur pour résoudre un problème ou atteindre un objectif.
 - (B) Une condition à satisfaire ou capacité à fournir par un composant du système de façon à honorer la fonction.
 - (C) Une représentation documentée d'une condition ou capacité comme définies selon (A) et (B).
- Outils : tableau Excel, document texte.

Formaliser le besoin

Approche IEEE (norme 830-1998)

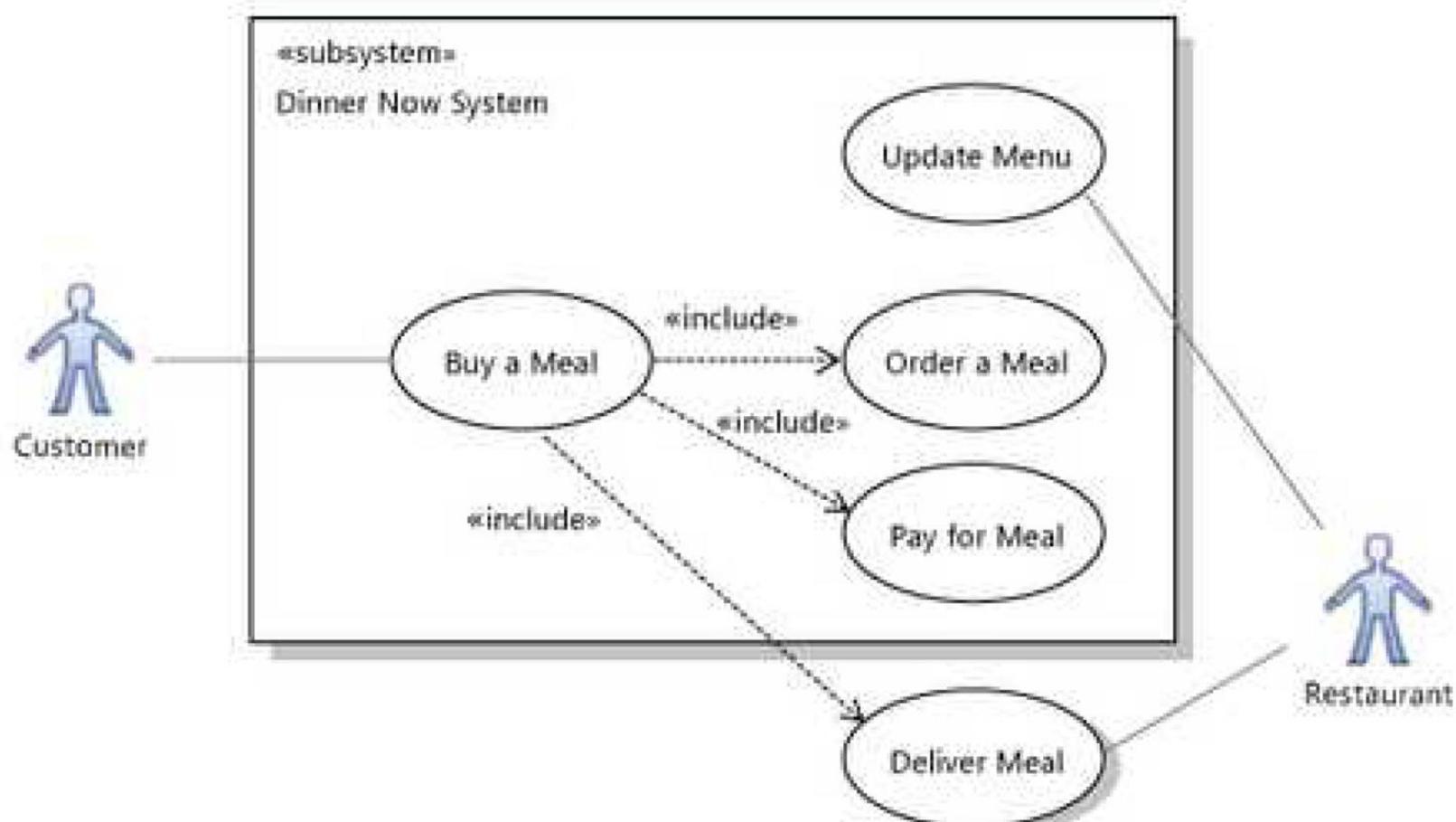
Identifiant	Exigences	Type	Priorité	Risque	Coût estimé	Status
		F	1	1	1	Fait
F01	Le système permet d'indexer des portions de documents source en associant le type de l'information.	F	1	1	1	Fait
F02	Le système permet d'associer aux produits des remarques destinées aux équipes éditoriales.	F	3	3	2	A faire
F03	Le système permet d'indiquer si un produit est commercialisé ou non.	F	2	3	3	A faire
F04	Le système permet d'associer aux produits un prix public TTC et un prix fabricant HT.	F	1	3	3	Fait
F05	Le système permet la saisie d'un même prix avec des dates d'effet différentes.	F	1	2	3	Fait
F06	Le système permet de référencer les produits au sein de la classification standard en vigueur.	F	2	3	2	En cours
F07	Le système permet un accès unique aux différentes applications au travers de différents onglets.	F	1	3	1	Fait
NF01	La recherche d'un produit par son identifiant doit aboutir en moins de 5 secondes.	NF	3	1	3	A faire
NF02	L'accès à chaque application n'est autorisé qu'aux utilisateurs ayant les droits.	NF	2	2	2	A faire

- Inconvénient : travail fastidieux et consommateur de temps – focalisation sur la description du produit et difficulté pour prioriser les exigences.

Formaliser le besoin Cas d'utilisation UML

- Technique de formalisation dynamique (recommandée dans UP).
- Description fonctionnelle des échanges entre le système à développer et les acteurs externes (utilisateurs ou autre).
- Forme : textuelle ou diagramme (+ séquence)
- NB. : préférer la forme textuelle qui ne nécessite aucune formation pour le comprendre et le valider.

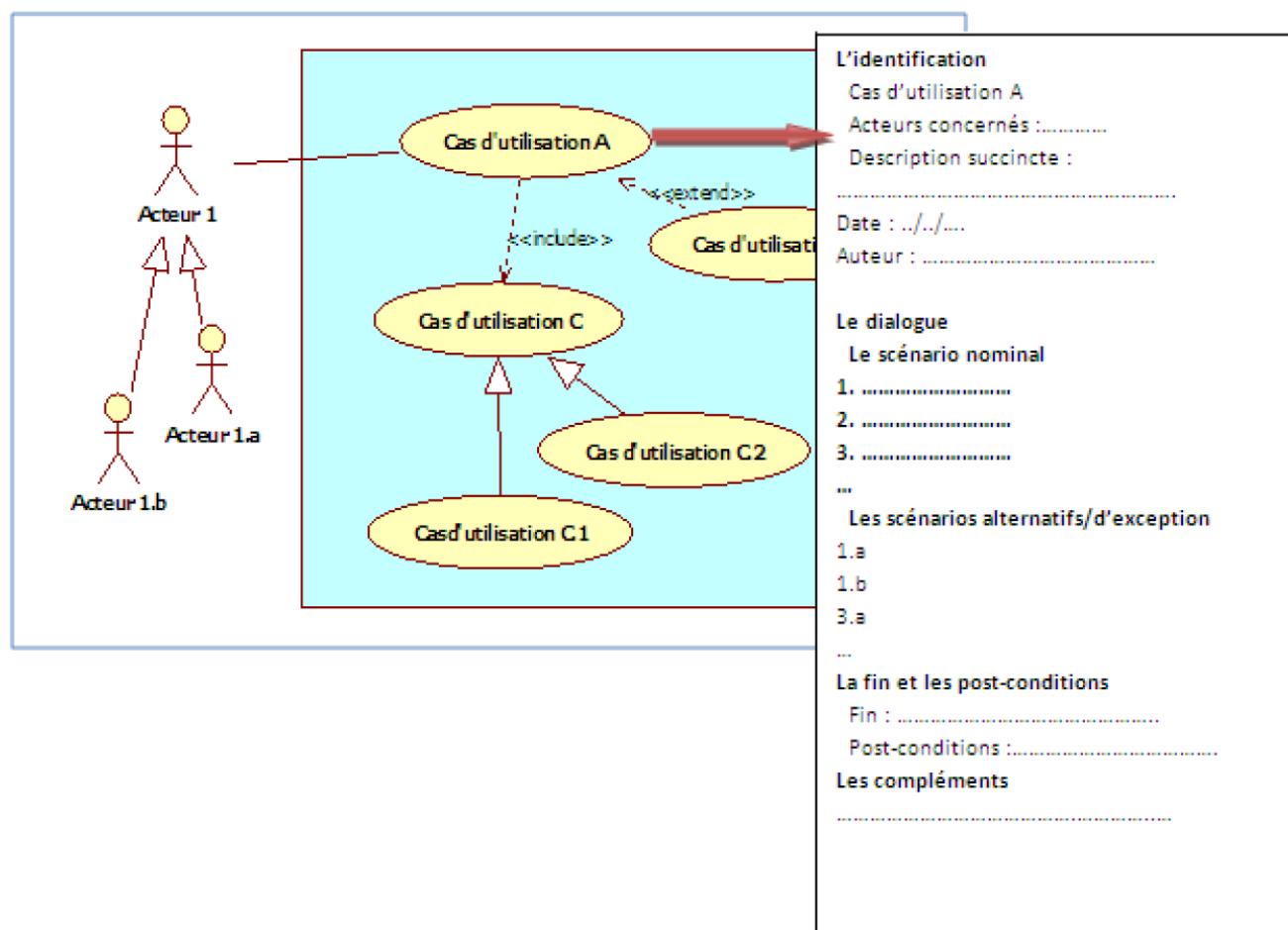
Formaliser le besoin Cas d'utilisation UML (2)



Formaliser le besoin

Cas d'utilisation UML - forme textuelle

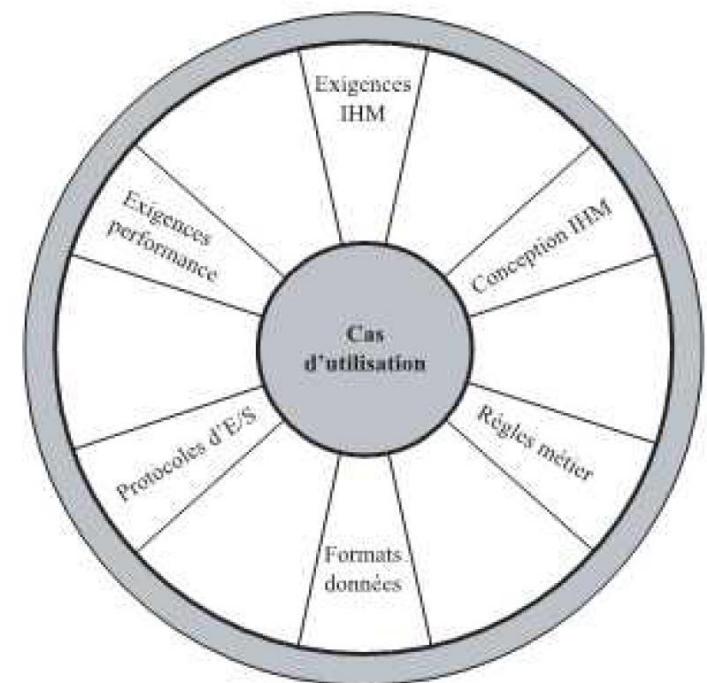
- La forme textuelle est recommandée pour plus de détails



Formaliser le besoin Cas d'utilisation UML (4)

- Moyeu-rayons des exigences : un UC ne décrit que les exigences fonctionnelles, qu'il faudra compléter avec d'autres exigences : interfaces externes, règles métier, formats de données, performance...

On considère le cas d'utilisation comme le « moyeu d'une roue » et les autres types d'exigences comme « des rayons partant dans des directions différentes »



Formaliser le besoin User Stories



- Exigence du système à développer (1 ou 2 phrases).
- Granularité suffisante pour estimer le coût et l'entièvre réalisation au cours d'une itération.
- Formalisme non imposé (pas de modèle).
- En tant que je dois pouvoir afin de ...
- **Avantages :**
 - - pas de détails superflus
 - - comportent des notes, des cas de tests,...
 - - facilitent la communication et l'évolution du scénario.

Formaliser le besoin User Stories (2)

- Une user story doit être **INVEST** :
 - Independant (indépendante)
 - Negotiable (négociable)
 - Valuable (Verticale, ayant de la valeur)
 - Estimatable (estimable)
 - Scalable (small sized) (suffisamment petite)
 - Testable (testable)
- **Inconvénient** : le client doit être à proximité et disponible
- Ensemble des users stories => Product Backlog.

Formaliser le besoin User Stories - INVEST



- **Indépendante** : lorsque le client peut en toute liberté décider de l'ordre dans lequel les scénarios (story) sont implémentés sans qu'interviennent des contraintes techniques.
Si des stories sont fortement liées, alors il faut les combiner.
- Bon exemple : En tant que client, je veux consulter la liste des factures émises.
- Mauvais exemple : En tant que client, je veux créer la liste des factures.

Formaliser le besoin

User Stories - INVEST (2)

- **Négociable** : L'équipe de développement n'est pas contrainte par la manière dont sera implémenté le scénario : Elle a la latitude d'imaginer une solution efficace.
- Bon exemple : En tant que client, je peux connaître le montant total des factures impayées.
- Mauvais exemple : En tant que client, je veux, lorsque je clique sur le bouton «calculer» une ligne s'ajoute et on affiche sur cette ligne le montant total des factures impayées.

Formaliser le besoin User Stories - INVEST (3)

- **Verticale** : une story décrit une fonctionnalité complète de l'application dont le client apprécie l'intérêt.
- Bon exemple : Un bon découpage qui fait apparaître les scénarios distincts rendus par le service de facturation :
 - Consulter la liste des factures émises.
 - Afficher la liste des factures ordonnées par date.
 - Afficher la liste des factures par adresse de livraison.
 - Consulter une facture client ...
- Mauvais exemple : Créer la base de données pour le module de facturation. Créer l'interface graphique pour la facturation.

Formaliser le besoin User Stories - INVEST (4)

- **Estimée** : pour qu'une story puissent servir de base à la planification, on doit connaître au moins une estimation du coût d'implémentation.
- Bon exemple : Implémenter les règles métier R1, et R2
 - R1 : si le nom de l'utilisateur existe déjà, lorsque l'acheteur en ligne essaye de créer son compte, alors le message «ce compte existe déjà » doit être affiché.
 - R2, un rabais de 5% est octroyé pour tous les clients dont le montant total de la facture dépasse les 1000e.
- Mauvais exemple : Garantir le respect des règles de gestion en vigueur (annexe jointe).

Formaliser le besoin User Stories - INVEST (5)

- **Suffisamment petite** (Size appropriately) : pour planifier sa réalisation un scénario doit être réalisable en peu de temps pour que l'équipe de projet puisse en planifier plusieurs dans un même sprint (itération)
- **Mauvais exemple** : réaliser le module de facturation
 - il faudra découper–
- **Bon exemple** : Un bon découpage qui fait apparaître les scénarios distincts rendus par le service de facturation :
 - Consulter la liste des factures émises.
 - Afficher la liste des factures ordonnées par date.
 - Afficher la liste des factures par adresse de livraison.
 - Consulter une facture client ...

Formaliser le besoin

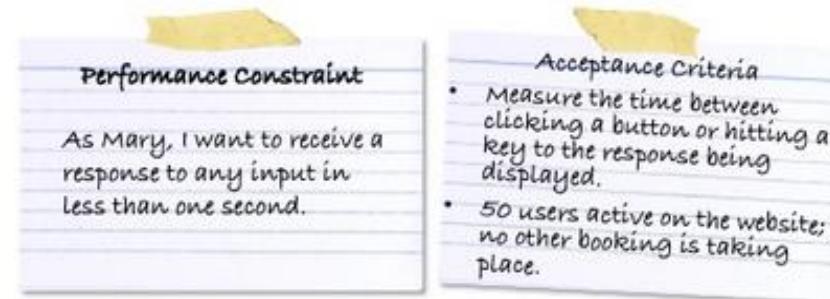
User Stories - INVEST (6)

• **Testable** : planifier un test d'acceptation est un excellent moyen pour vérifier que tout le monde a bien compris le scénario ou la story.

• **User Story** :

En tant que client, je veux connaître la liste des factures par date.

• **Test d'acceptation** : afficher la liste des factures ordonnées par la date de facturation



Formaliser le besoin

User Stories - Tests d'acceptation



[titre] L'utilisateur peut rechercher un emploi
Le programme sera écrit en Java (techno !)

[description; 1 ou 2 phrases support à la conversation]

L'utilisateur peut voir les informations de chaque emploi qui correspond à sa recherche.

[détails, questions, Les décisions prises deviennent des tests.]

- Marco suggère d'afficher la description, le salaire et le lieu.
- Les cartes de paiement Discover sont-elles acceptées ?
- On accepte les cartes Visa, Mastercard et American Express+ (c'est 1 test : voir au verso)

[risque technique]

2

[coût, en story points]

4

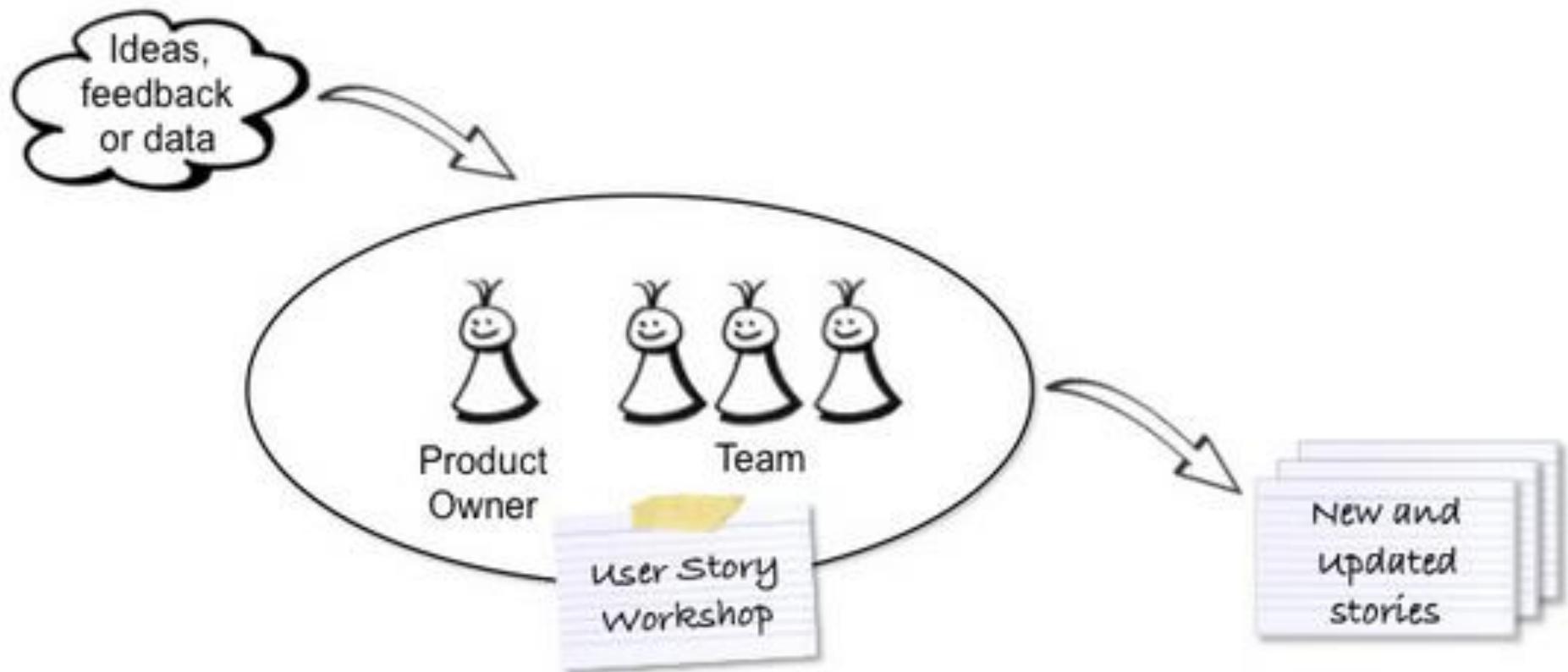
[expectations, acceptance tests. Si tous sont ok la story est achevée]

- Tester avec une description de job vide.
- Tester avec une description de job très longue.
- Tester sans indiquer de salaire.
- Tester avec un salaire à 6 chiffres.
- Tester avec Visa, MasterCard et American Express → succès !
- Tester avec Diner's Club → échec !

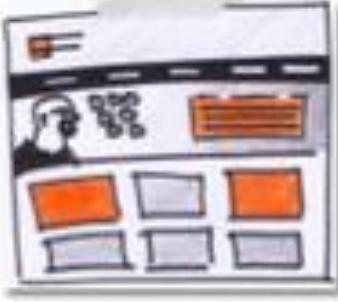
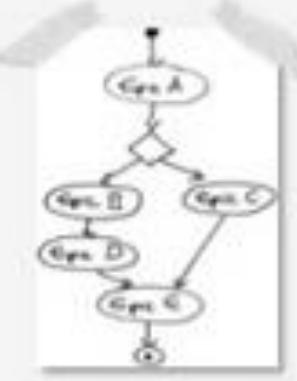
[une bonne story doit être INVEST (Bill Wake, 2003)]

- **Independent** (entre elles, revoir le découpage sinon)
- **Negotiable** (se n'est pas un contrat écrit !)
- **Valuable** (par l'utilisateur, l'acheteur, le client : pour le planning)
- **Estimatable** (par le développeur)
- **Small** (idéalement pas plus de qq jours, excepté les epics)
- **Testable** (si la story est non testable → comment savoir si finie ?)

Formaliser le besoin User Stories - workshop



Formaliser le besoin User Stories + high level design

Story Area	Constraint Area	Model Area
<p>1. Story 2. Story 3. Story</p> <p>4. Story 5. Story</p> <p>Epic A Epic B Epic C</p> <p>Epic D Epic E</p> <p>Epic F Epic G</p>	<p>Performance Constraint</p>  <p>Robustness Constraint</p> <p>Int.coup. Constraint</p>	

Mettre en place des critères d'acceptation

Atelier “three amigos”



- L'atelier "three amigos" aide à l'écriture des critères d'acceptation sous la forme de **BDD (Behaviour Driven Development)**.
- **Acteurs** : 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.
- **Procédé** : 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.
- **Résultat** : 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)

Tests d'acceptation (automatisation)

- La syntaxe utilisée, appelée désormais **langage Gherkin**, est de plus en plus reconnue par des frameworks (tels que **jBehave** ou **Cucumber**) permettant d'automatiser ces tests fonctionnels.
- Il 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.
- Exemple :
GIVEN un utilisateur avec une carte de n° 1234567890, valide jusqu'en 01/16 et de code sécurité 123,
WHEN il saisit la carte 1234567890
 - **AND** la date de validité 01/16
 - **AND** le code de sécurité 123,
THEN, le paiement est accepté
 - **AND** il est redirigé vers la page de confirmation du paiement.

Formaliser le besoin

User Stories vs Use Case

- Le choix de la technique dépend du contexte et de la méthode adoptée.
- Insister sur la formalisation si les équipes sont dispersées géographiquement => Use Case.
- Si le client est disponible et le nombre d'interlocuteurs réduit => User Stories.

Formaliser le besoin

User Stories vs Use Case

User Story	Use Case
Brève description d'une fonctionnalité (vue utilisateur)	Séquence d'actions (interactions entre les acteurs du système)
Format écrit court, échanges de vive voix	Format écrit très riche en informations. Peu de place à l'oral
Peut être une partie d'un use case (scénario principal ou une alternative)	La somme d'un scénario principal et de diverses alternatives (variations, cas d'erreur,...)
Spécification utilisée pour l'estimation de la planification	Spécification seulement
Émergence rapide au travers d'ateliers collaboratifs	Long travail d'analyse et de formalisation
Grande lisibilité (simple)	Peut souvent manquer de lisibilité (volume)
Mode oral et collaboratif	Mode écrit et souvent distant
Facile à maintenir (format riche, court, indépendant)	Plus difficile à maintenir (documents volumineux)
Implémentée et testée en une itération obligatoirement	Implémenté et testé en plusieurs itérations
Écrite facilement par un utilisateur ou un client (accompagné dans sa démarche)	Souvent rédigé par des user proxies (représentants de l'utilisateur : AMOA, analyste,...). Rarement par le client final
Contient des tests d'acceptation (implication des testeurs)	Ne contient pas les cas de test (pas d'implication des testeurs)
Difficile à lier les unes aux autres. Absence de vue globale	Liaison et vision globale facilitées : au travers du diagramme des cas d'utilisation qui relie les use cases entre eux
Associée historiquement à l'eXtreme Programming et aux méthodes agiles	Associé historiquement au processus unité
Auteur de référence : Mike Cohn	Auteur de référence : Alistair Cockburn

Product Backlog (PB)

- Référentiel regroupant l'ensemble des besoins/exigences ou des livrables à réaliser.
- PB = liste de Product Backlog Items (PBI) hiérarchisés **par le client** en fonction de leur valeur ajoutée.
- Il est évolutif (recueil dynamique au fil du projet) :
 - référence les user stories annulées, subdivisées,...
 - recense les évolutions
- Le PB est sous la responsabilité du seul représentant du client, le Product Owner (Scrum).

Product Backlog (2)

ID	Scénario utilisateur	Test d'acceptation	Commentaires	Points	Coût	Jours
HOP5	<p><i>En tant que user</i>  <i>Je peux view a set of simple screen shots</i> Completed sprint 5</p>	a) Lightbox appears with ability to browse through gallery of screen shots	Assumed use of JQuery Lightbox	3	800 €	1.0
HOP2	<p><i>En tant que admin</i> Accepted sprint 1</p> <p><i>Je peux track users who arrive on the home page via a campaign URL</i> <i>Pour pouvoir measure the effectiveness of my campaigns</i></p>	a) Support for tracking code in the format ?campaign=[code] b) Tracking code and number of visits to be stored in the database		3	800 €	1.0
HOP3	<p><i>En tant que user</i> Accepted sprint 1</p> <p><i>Je peux see the latest tweets from the company</i> <i>Pour pouvoir keep up to date with the latest news and comment for the company</i></p>	a) Tweets are automatically pulled from @company See http://twitter.com/#/company b) The most recent 5 tweets should be visible c) Tweets must not be links, they should only be shown as plain text		8	2 133 €	2.7
HOP4	<p><i>En tant que investor</i> Accepted sprint 2</p> <p><i>Je peux see information on the company and its products</i> <i>Pour pouvoir get an idea about what the company does and its relevance</i></p>	a) Short about text to be displayed b) All products from the products section to be listed on the home page with roll over screen shots and product info c) Products will deep link to the products section of the website	All design has been completed by Jason. HTML and templates to be built	8	2 133 €	2.7
Add story		Total du thème 'Home page'		22.0 points / 5 867 € / 7.3 jours		

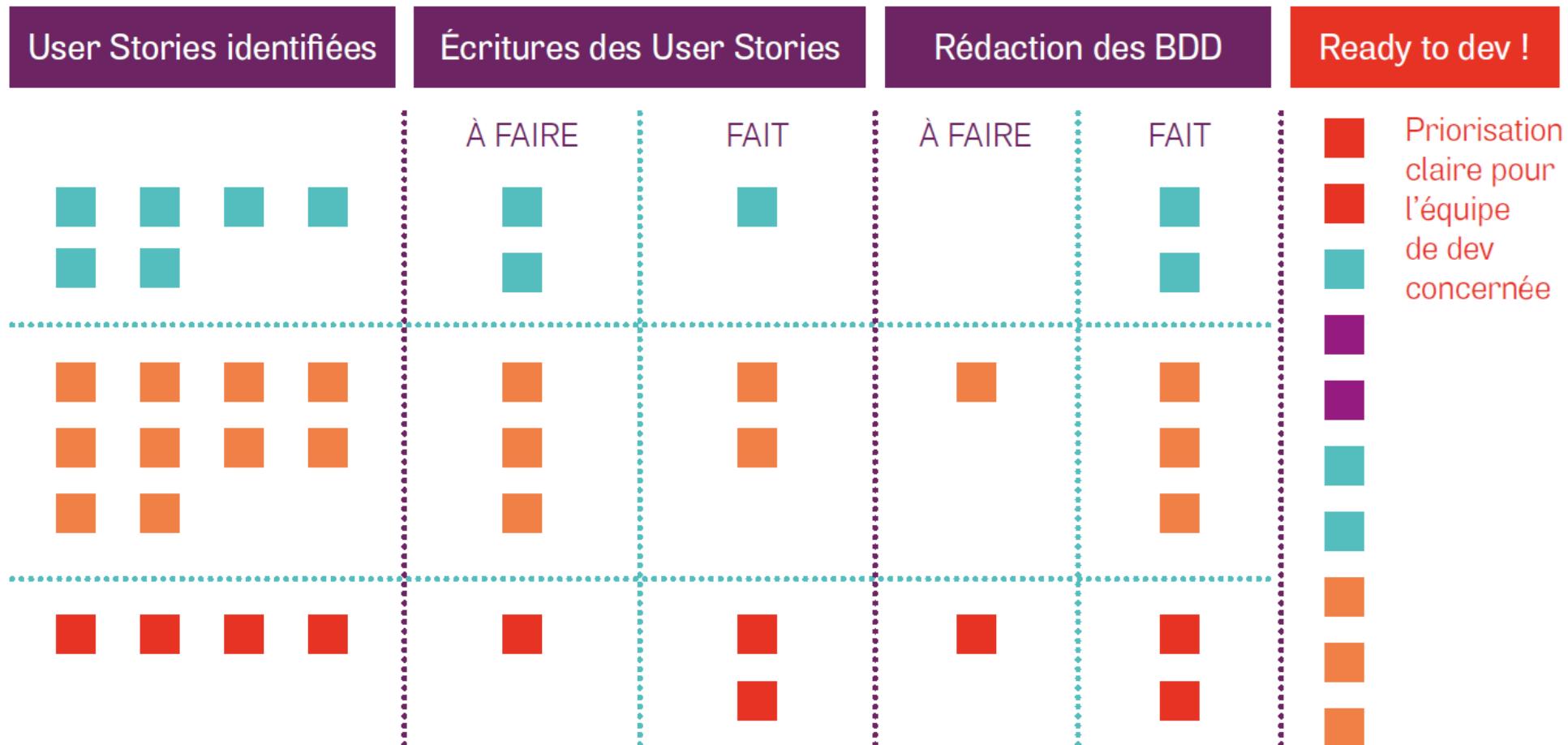
- Outils : Jira, IceScrum, ScrumDo, ScrumWise, easyBackLog, Agilefant,...

- Démo : easyBackLog

Organiser la construction du produit



KANBAN DU READY



Comment estimer et prioriser les fonctionnalités ?



Techniques d'estimation

2 techniques :

- **Ideal days** : estimation en jour homme (charge).
- **Story points** : on calcule un nombre de points abstrait (complexité).
A vaut 5 points et B vaut 10 points
=> B nécessite le double d'effort pour être développée.
- **Taille globale = somme des points des user stories**

Techniques d'estimation

Méthode des ideal days

- **Temps idéal (ideal time)** = temps nécessaire à la réalisation d'une tâche sans interruption de travail, avec toutes les ressources requises.
- **Temps écoulé (elapsed time)** = temps réellement constaté sur une horloge ou un calendrier durant la réalisation de la tâche.
L'estimation de la taille du projet en temps idéal suppose que l'on évalue le nombre d'ideal days pour développer toutes les user stories, sans qu'aucune interruption ne vienne perturber le travail.
Avec la vélocité de l'équipe, on est en mesure de déduire la durée du projet, comme avec la technique des story points.

Cette technique fonctionne sans problème avec une équipe qui se connaît bien et qui a l'habitude de travailler ensemble ; sa vertu est que l'on accepte les « surcharges pondérales » non prévues, ces tâches invisibles qui allongent systématiquement le délai et qui ne sont pas toujours prises en compte dans une planification classique.

Techniques d'estimation

Méthode des story points

Comment évaluer le nombre de points par story ?

- On sélectionne la story qui semble la plus petite et on lui affecte un poids de 1. On estime le poids des autres en leur affectant un nombre allant de 1 à 10 par comparaison avec la première. On peut se référer à la suite de Fibonacci (1,2,3,5,8,13,...).
 - Il faut ensuite convertir ce nombre abstrait en jour ; il faut connaître la productivité de l'équipe (vitesse).
- Vitesse = somme des story points qu'une équipe est capable de développer durant une itération (charge de travail maximale).**
- A partir du calcul de la vitesse de l'équipe, **hypothétique au début** puis plus réaliste à l'issue d'une première itération, on recalibre cette vitesse ; on peut déduire le nombre d'itérations et donc la durée du projet, en divisant la taille globale par la vitesse.

Techniques d'estimation

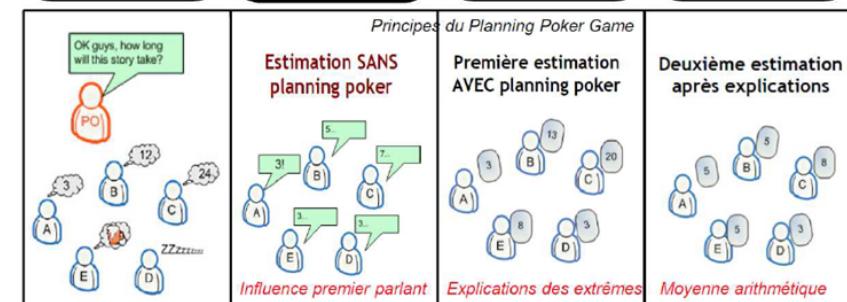
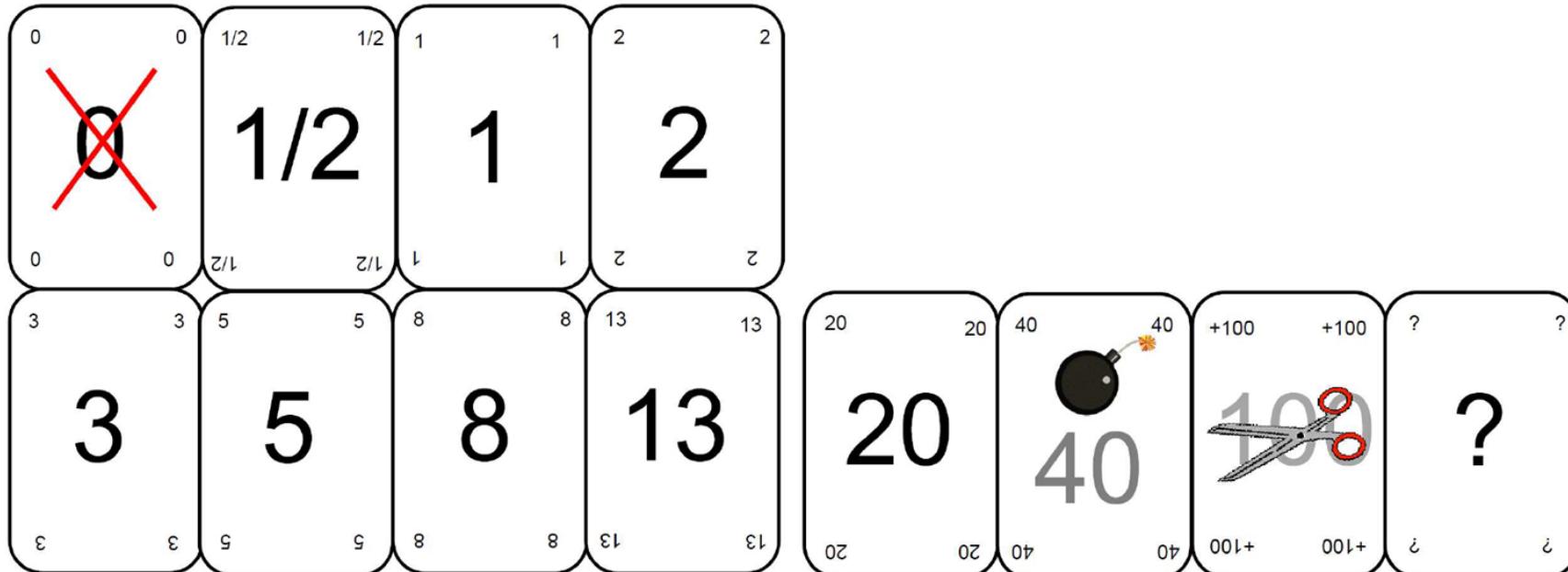
Méthode des story points (2)

- **Exemple :** Si la taille du projet est estimée à 100 story points et que la vitesse de l'équipe est estimée à 10 points pour une itération de deux semaines, on peut en déduire que le projet durera dix itérations de deux semaines, soit vingt semaines.

Techniques d'estimation

Méthode des story points (3)

- Outil d'estimation : Planning poker



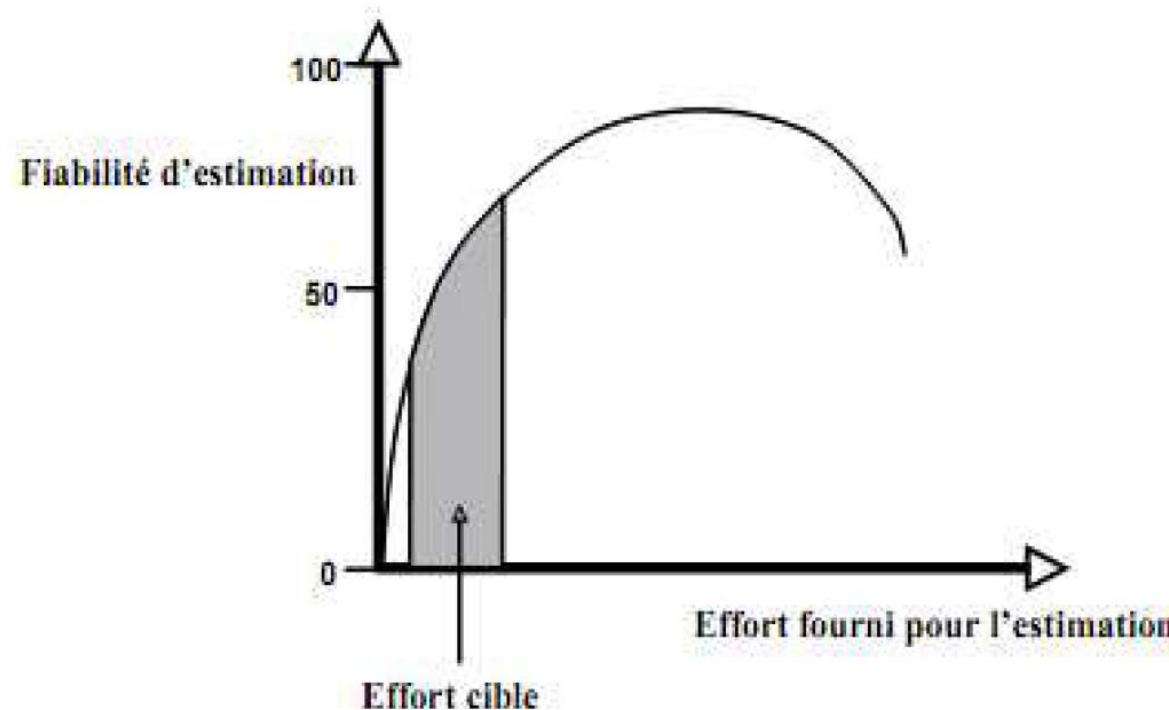
Fiabilisation de l'estimation

- Les estimations résultent d'une démarche plus ou moins empirique qui s'appuie sur différentes techniques, selon l'approche retenue. Elles forment la base pour des discussions, des négociations, des compromis à faire entre contenu, coût et délai.
- **Mais notre engagement est subordonné à l'exactitude des estimations**, qui dépendent de la qualité des données d'entrée (clarté du besoin, vélocité de l'équipe, facteurs de risques...), des données capitalisées et comprenant une part inéluctable de subjectivité. L'estimation exacte ne sera définitivement connue qu'à la fin du projet !

Fiabilisation de l'estimation (2)

Faut-il consacrer un effort important aux estimations ?

- On constate qu'il n'y a pas de corrélation systématique entre le temps consommé sur les estimations et la qualité de ces estimations.



Fiabilisation de l'estimation (3)

- En acceptant l'incertitude, **les équipes agiles privilégient l'effort pour produire régulièrement des versions intermédiaires de l'application**, et ainsi fiabiliser leurs estimations, basées sur le retour d'expérience. **Elles considèrent l'estimation initiale comme une enveloppe, à partir de laquelle on peut se mettre d'accord sur un coût et un délai** ; sur la base de cette enveloppe, on adapte le contenu au fil des itérations en fonction de ce qu'il est pertinent de développer.
- On peut tenter de fiabiliser l'estimation, en adoptant une démarche collaborative
- **Wide Band Delphi (loi bêta) : $E = [P + (4 \times I) + O] / 6$**
P = estimation pessimiste,
I = estimation intermédiaire, la plus probable
O = estimation optimiste la plus représentative et consensuelle.

Hiérarchiser les besoins

- Quel est le bénéfice financier à développer cette fonctionnalité ?
- Quel est le coût de développement ? De maintenance ? De formation des utilisateurs ? Quel est le coût d'un changement ?
- L'implémentation de cette fonctionnalité nous permet-elle d'apprendre ou de développer de nouvelles compétences ?
- Cette fonctionnalité nous expose-t-elle à davantage de risques ?
- Cette fonctionnalité va-t-elle impacter le processus métier ?
- Cette fonctionnalité va-t-elle améliorer la productivité de mon personnel ?
- Cette fonctionnalité va-t-elle susciter la frustration ou la résistance ?
- Quel est le préjudice si cette fonctionnalité n'est pas implémentée ?

Hiérarchiser le besoin Bénéfice financier attendu

- Facile à évaluer pour un produit commercial.
- ROI (Return Of Investment) difficile à chiffrer pour un produit interne :
 - Quantifier le taux d'insatisfaction ou le préjudice subi si une fonctionnalité n'est pas implémentée.
 - Évaluer l'amélioration de la productivité (gains de temps, réductions de coûts directs)

Hiérarchiser le besoin Coût de développement estimé



- Facteur déterminant
- Il faut implémenter les fonctionnalités coûteuses le plus tard possible (meilleure maîtrise du produit, meilleures visibilité / productivité).
- S'il y a des reports, tenir compte du ratio coût / valeur.

Hiérarchiser le besoin Opportunité d'apprentissage pour l'équipe

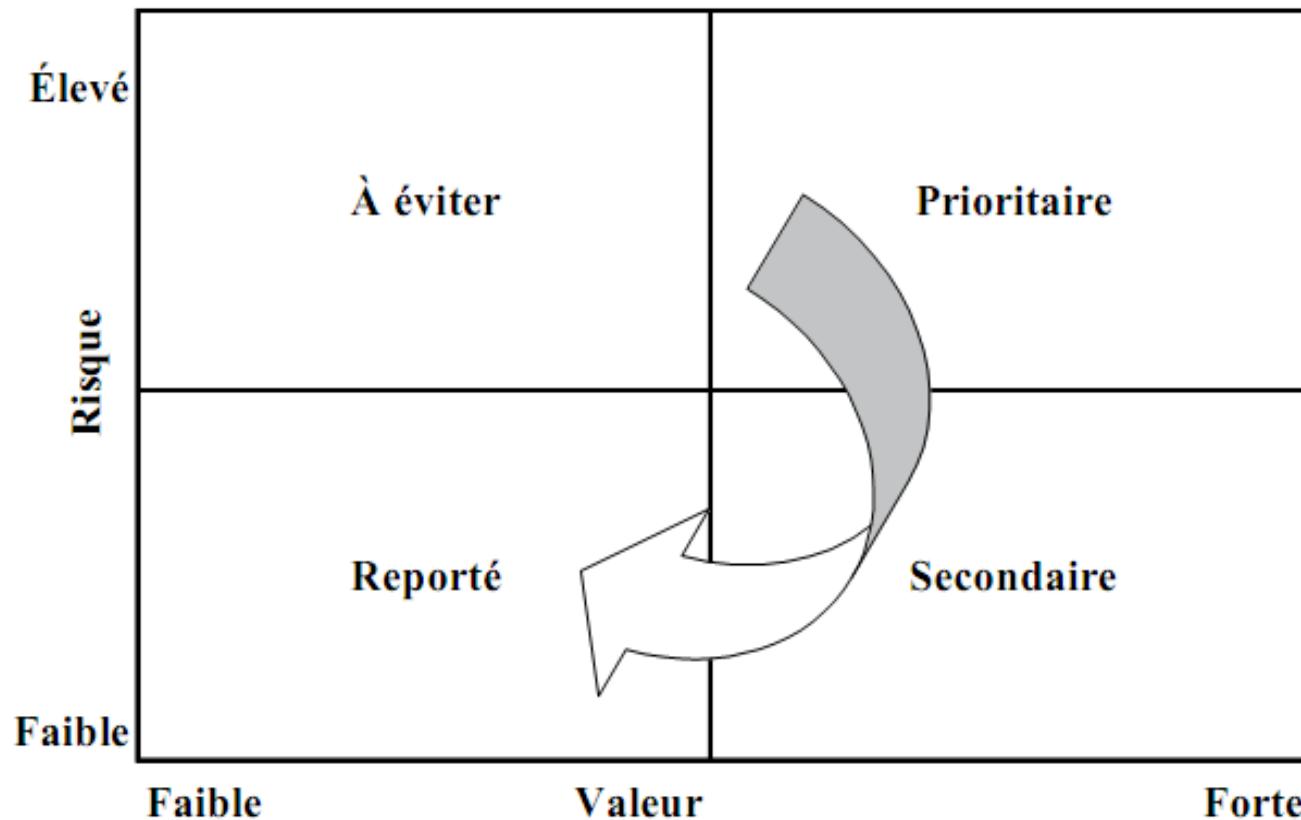
- Nécessité de montée en compétences sur de nombreux projets (technologies, domaine métier, bonnes pratiques).
- Apprentissage rapide
- => équipe plus productive
- => coûts et risques réduits
- Les fonctionnalités qui lèvent les incertitudes liées à la découverte sont donc prioritaires.

Hiérarchiser le besoin Risque de développement

- Faut-il implémenter les fonctionnalités les plus risquées, au détriment de la valeur ajoutée pour le client ?
- Ou, faut-il se focaliser sur la satisfaction du client en reportant la confrontation au risque ?
- Recommandations :
- Les fonctionnalités à forte valeur ajoutée sont prioritaires ;
- Celles qui sont le plus risquées sont développées avant celles qui sont le moins risquées ;
- Celles qui ont peu de valeur et sont très risquées sont reportées voire éliminées.

Hiérarchiser le besoin Risque de développement (2)

Hiérarchisation des fonctionnalités en fonction des risques et de la valeur



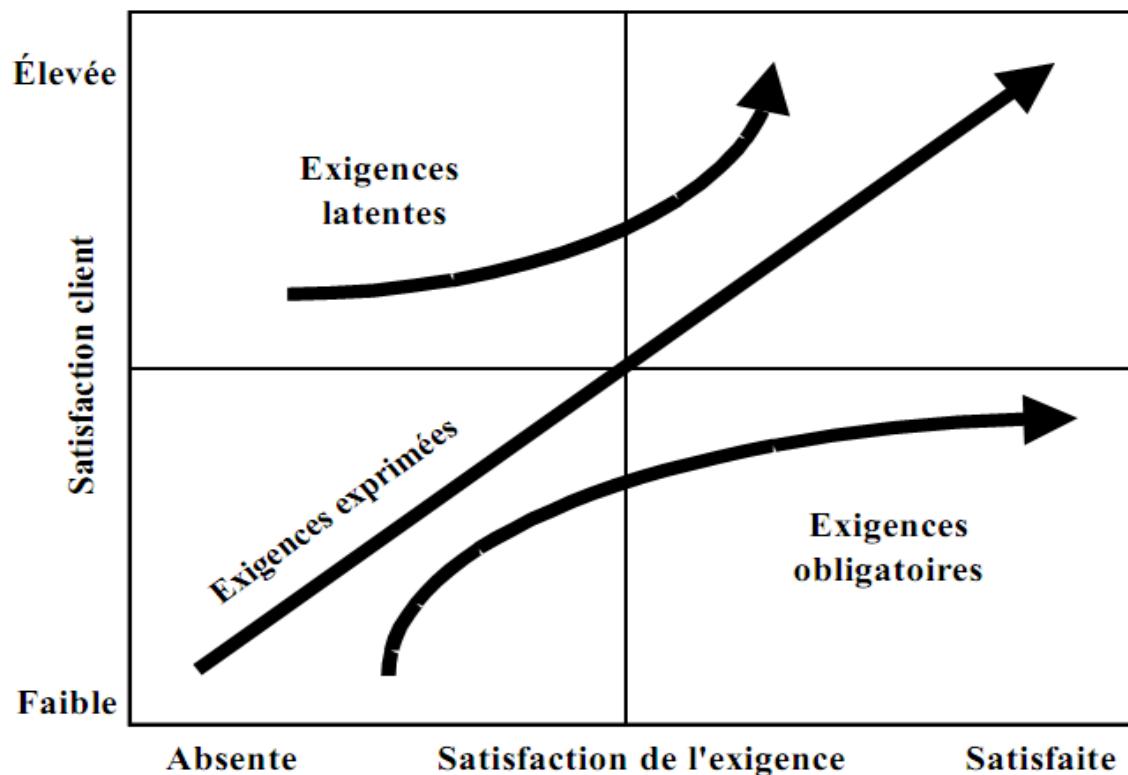
Hiérarchiser le besoin Degré de satisfaction du client

- Comment livrer un produit satisfaisant si l'on n'a pas cartographié ce qui est considéré comme satisfaisant par le client lui-même ?
- 3 modèles :
- - Modèle de Kano
- - Méthode des poids relatifs
- - Méthode MOSCOW

Degré de satisfaction du client

Modèle de Kano

- Classification des exigences en 3 catégories :
- obligatoires, exprimées et latentes.
- Mesure de la corrélation avec la satisfaction du client.



Degré de satisfaction du client

Modèle de Kano - Formalisation

- Enquête auprès d'un échantillon représentatifs d'utilisateurs pour obtenir une échelle de valeur.
- 2 questions, 5 réponses possibles (format Excel)

Fonctionnalité n	Si cette fonctionnalité était présente, que ressentiriez-vous ?	
	« Ça me ferait plaisir »	
	« Ce serait un minimum »	X
	« Je n'ai pas d'avis »	
	« Je l'accepterais »	
	« Ca me dérangerait »	

Fonctionnalité n	Si cette fonctionnalité n'était pas présente, que ressentiriez-vous ?	
	« Ça me ferait plaisir »	
	« Ce serait un minimum »	
	« Je n'ai pas d'avis »	
	« Je l'accepterais »	X
	« Ca me dérangerait »	

Question fonctionnelle	Question dysfonctionnelle				
	Plaisir	Minimum	Neutre	Acceptation	Dérangement
Plaisir	?	L	L	L	E
Minimum	R	I	I	I	O
Neutre	R	I	I	I	O
Acceptation	R	I	I	I	O
Dérangement	R	R	R	R	?

O Obligatoire R "Reverse" (Inversée)
E Exprimée ? Incertaine
L Latente I Indifférente

Degré de satisfaction du client

Méthode MOSCOW



Échelle de hiérarchisation des besoins

- M « Must have » (Indispensable)
- S « Should have » (Souhaitable)
- C « Could have » (Possible)
- W « Want to have but Won't have » (souhaité mais non réalisable pour le moment donc Éliminé)
- Les O ne sont que des objets de liaison pour les acronymes
 - Format : papier ou Excel

Degré de satisfaction du client

Méthode MOSCOW - procédé

- Liste toutes les Users Stories à prioriser.
- Demandez au Product Owner quelles Users Stories doivent absolument être disponibles (liste de "MUST" , Doit être réalisé).
- Refaites un passage sur ces Users Stories : Il arrive souvent que le PO veuille tout dans 1 semaine, identifiez celles qui ne sont pas si essentielles et mettez les comme SHOULD (Devrait être réalisé).
- Écartez les "MUST" et refaites un passage en listant les fonctionnalités qui devraient être en ligne juste après celle-ci (vous obtenez la liste des "SHOULD").
- Il ne vous reste que des "COULD" et "WON'T".
- Optionnel : demandez à identifier les "WON'T" (les Users Stories qui ne doivent pas être présentes), cela permet de voir les "COULD" (Pourrait être réalisé).
- Reclassez vos Users Stories dans l'ordre : MUST, SHOULD, COULD et WON'T et parcourrez la liste avec le Product Owner et faites les derniers ajustements.

Degré de satisfaction du client

Méthode des poids relatifs

- Relative weighting (Karl Weigers)
- Évaluer pour chaque item (exigence, story,...) :
 - le bénéfice relatif (A) de son implémentation, sur une échelle de 1 à 9 (1=peu de valeur, 9=bénéfice maximal)
 - le préjudice relatif (B) si la fonctionnalité n'est pas implémentée (1=faible préjudice, 9=préjudice maximal)
- Valeur d'une exigence = bénéfice + préjudice. ($C=A+B$)
- Valeur globale = valeur exigence / valeur totale. ($D= C/C'$)
- Estimation du coût de développement (E)
- Poids = coût exigence / coût global. ($F=E/E'$)
- Priorité = %valeur / %coût. ($Priorité=D/F$)

Degré de satisfaction du client

Méthode des poids relatifs (2)

Exigence	Bénéfice relatif A	Préjudice relatif B	Valeur totale C	% D	Valeur	Coût estimé E	% F	Coût	Priorité
Exigence A	5	2	7	22,5		20	27,8		0,81
Exigence B	8	9	17	55		45	62,5		0,88
Exigence C	1	6	7	22,5		7	9,7		2,31
Total	14	17	31	100		72	100		
			C'			E'			

On peut inclure d'autres critères en définissant un poids pour chacun :
 (source : Demand Metric)

	Strategic Fit			Economic Impact			Feasibility			Total
Selection Criteria	Alignment with Goals	Market Positioning	Capabilities	Revenue Potential	Cost/Benefit	Low Cost	Technical Risk	Resources - Financial	Resources - People	Weight
Weighting Scale	15%	15%	10%	10%	15%	15%	10%	5%	5%	100%

Planification du projet

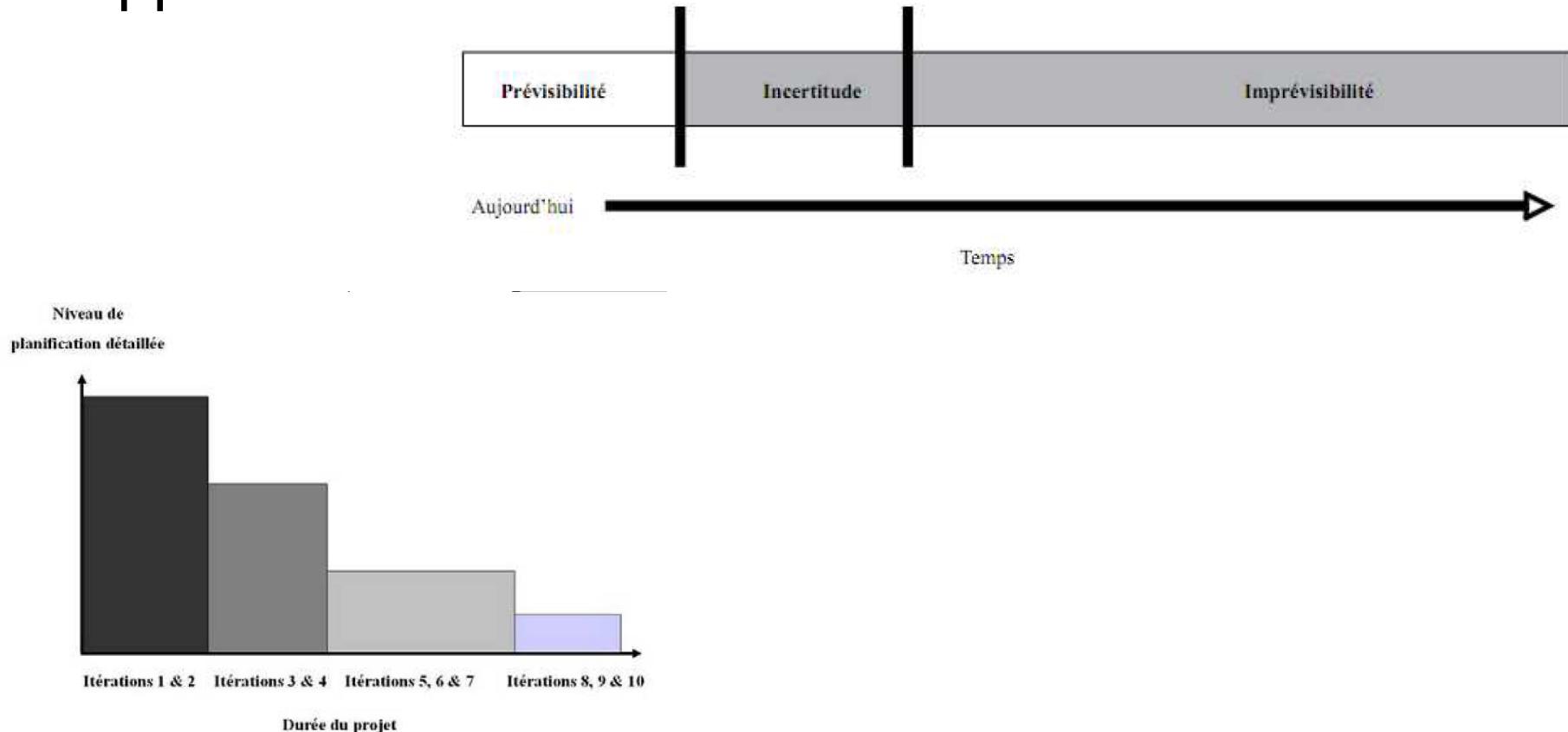


Planifier son projet

- Le plan doit avoir comme objectif la satisfaction du client.
- Ne pas confondre estimation et engagement !
- Pourquoi planifier ?
 - - Outil d'aide à la décision (investissement à engager, dépendance avec des activités ou objectifs commerciaux)
 - - Préparer les échéances principales. Le client peut programmer ses actions futures.
 - - Rassurer la hiérarchie !
 - - Outil de pilotage pour le suivi du projet

Approche adaptative

- Contrairement à un projet classique (approche prédictive, tout est figé dès le début), les méthodes agiles préconisent une approche adaptative (planifier au fil de l'eau)



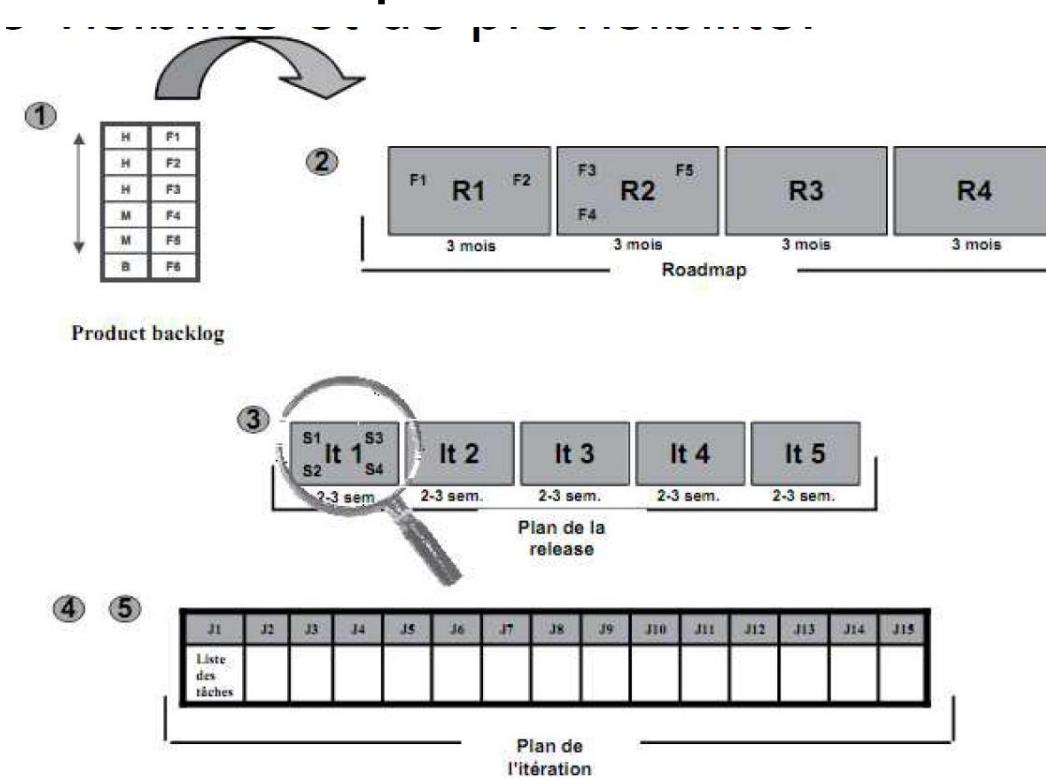
Planification Agile

5 niveaux de planification qui correspondent aux différentes étapes qui rythment le projet :

- Établissement de la vision.
- Fixation des jalons.
- Planification d'une release.
- Planification d'une itération.
- Planification quotidienne.

Planification Agile (2)

- Ne pas croire aux plannings détaillés en amont.
- S'attacher à avoir un niveau de détail adapté à notre niveau de visibilité et de prévisibilité.



Planification Agile

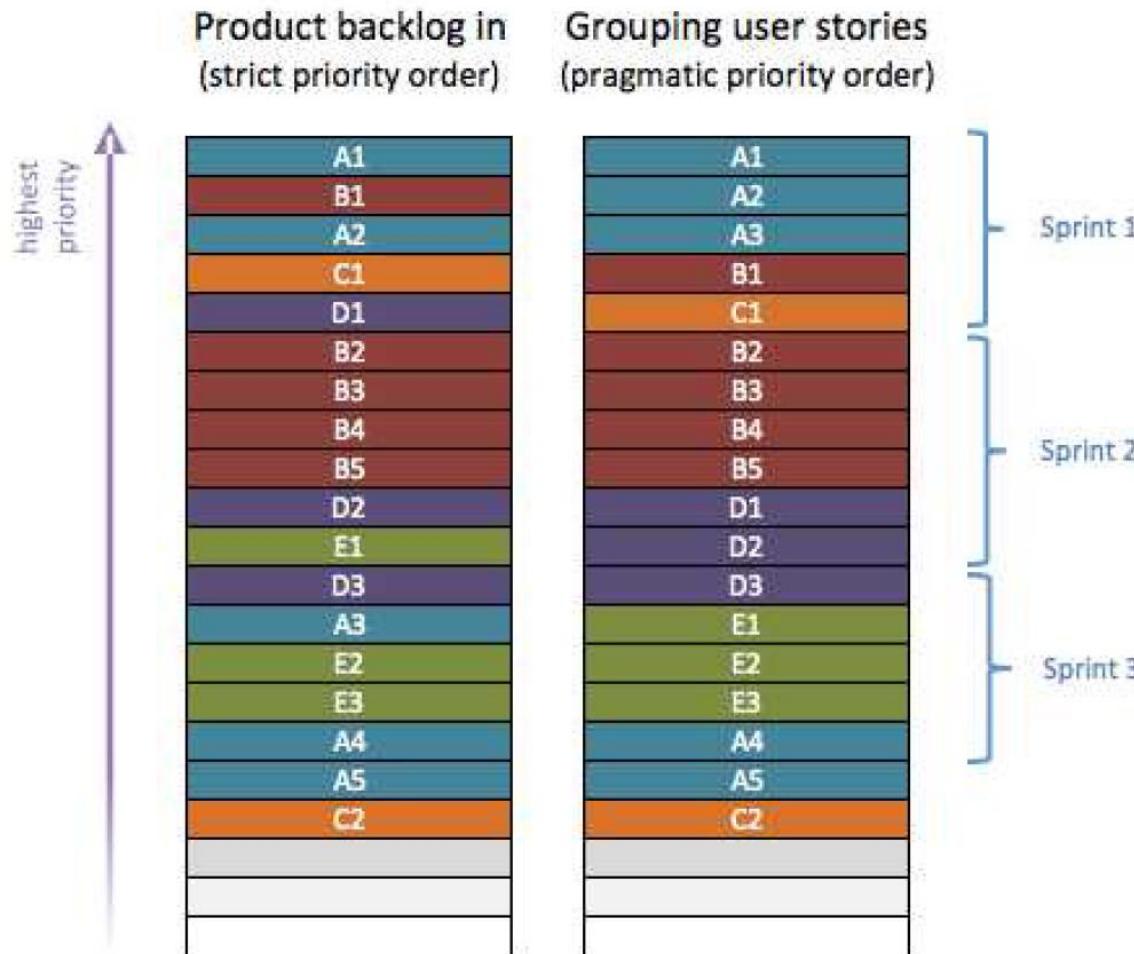
Vision du produit

- Niveau de planification global établi à partir de la vision du produit et du **Product Backlog** (priorisé par le client).

Backlog de produit - 215pts						
R1#1 Décomposer un	R1#1 Copie features	R1#1 Template Test	R1#1 Filtres sur les Tests	R1#1 Etude graphiques	R1#1 Messages erreur	
En tant que Product Owner, Je peux décomposer un élément du backlog en 2 Min de Fini	En tant que Product Owner, Je peux copier une feature de la liste dans le backlog de produit Fini	En tant que Product Owner, Je peux spécifier un test en utilisant le template BDD Afin de Fini	En tant que Product Owner, Je peux filtrer les tests Afin de Fini	En tant que ScrumMaster, Je peux lister tous les graphiques Afin de Fini	En tant que ScrumMaster, Je peux lister les messages Afin de Fini	
R1#2 Bug visuel cancellé...	R1#2 Graphe de vitesse	R1#2 Mettre à jour les	R1#2 Nouveau test à	R1#2 Copier un cas de	R1#2 Sortie d'un build	
En tant que N'importe qui, Je peux need to rollback. En cours	En tant que Stakeholder, Je peux produire un graphique avec la vitesse des sprints de En cours	Mettre à jour les droits en fonction des rôles En cours	En tant que Product Owner, Je peux créer un test en sélectionnant une story En cours	En tant que Product Owner, Je peux copier un cas de test Afin de gagner du temps dans la sélection de l'élément En cours	En tant que ScrumMaster, Je peux indiquer la sortie d'un nouveau build Afin de passer les En cours	
R1#2 Passage d'un test	R1#2 Boutons Valider et	R1#2 Expliciter le widge...	R1#3 Rapport build	R1#3 Consommation	R1#3 Agrandissement de	
En tant que Team Member, Je peux passer un test et indiquer succès ou échec, avec En cours	En tant que Product Owner, Je peux valider et continuer aussi pour les tests et pas seulement En cours	En tant que Product Owner, Je peux comprendre les chiffres Afin de En cours	En tant que Product Owner, Je peux savoir quels tests ont été passés et leur résultat pour Planifié	En tant que ScrumMaster, Je peux indiquer pour chaque sprint le nombre de jours Planifié	En tant que ScrumMaster, Je peux voir le détail de la note Tâche Afin de Planifié	
R1#3 Filtrer par feature	R1#3 Décomposer en	R1#4 Imprission story	Supprimer l'onglet ...	Focus dans le premi...	Navigation entre le...	
En tant que Product Owner, Je peux obtenir la liste des éléments du backlog associés à Planifié	En tant que Product Owner, Je peux décomposer un élément en plusieurs Afin de Planifié	En tant que Product Owner, Je peux imprimer des cartes à découper à partir du backlog Planifié	En tant que N'importe qui, Je peux accéder aux caractéristiques en cliquant sur Validé	En tant que N'importe qui, Je peux saisir directement sans avoir à positionner le curseur Estimé	En tant que Product Owner, Je peux naviguer en sélectionnant une feature sur le backlog ne Validé	
Plus de contenu dan...	Bugs admin	Page d'accueil selo...	Ouverture projet	Template de créatio...	Slack	
En tant que N'importe qui, Je peux agrandir la taille d'un widget, ce qui met plus de Validé	En tant que Administrateur, Je peux supprimer un user sans que ça plante et avoir des Estimé	En tant que Stakeholder, Je peux avoir une page d'accueil spécifique Afin de Estimé	En tant que Stakeholder, Je peux accéder directement à mon projet si je ne suis associé Estimé	As Team Member, I can Create template for tasks To Allow better use of icescrum UI Estimé	En tant que ScrumMaster, Je peux connaître le mou disponible pendant un sprint le Estimé	

Planification Agile

Vision du produit (priorisation)



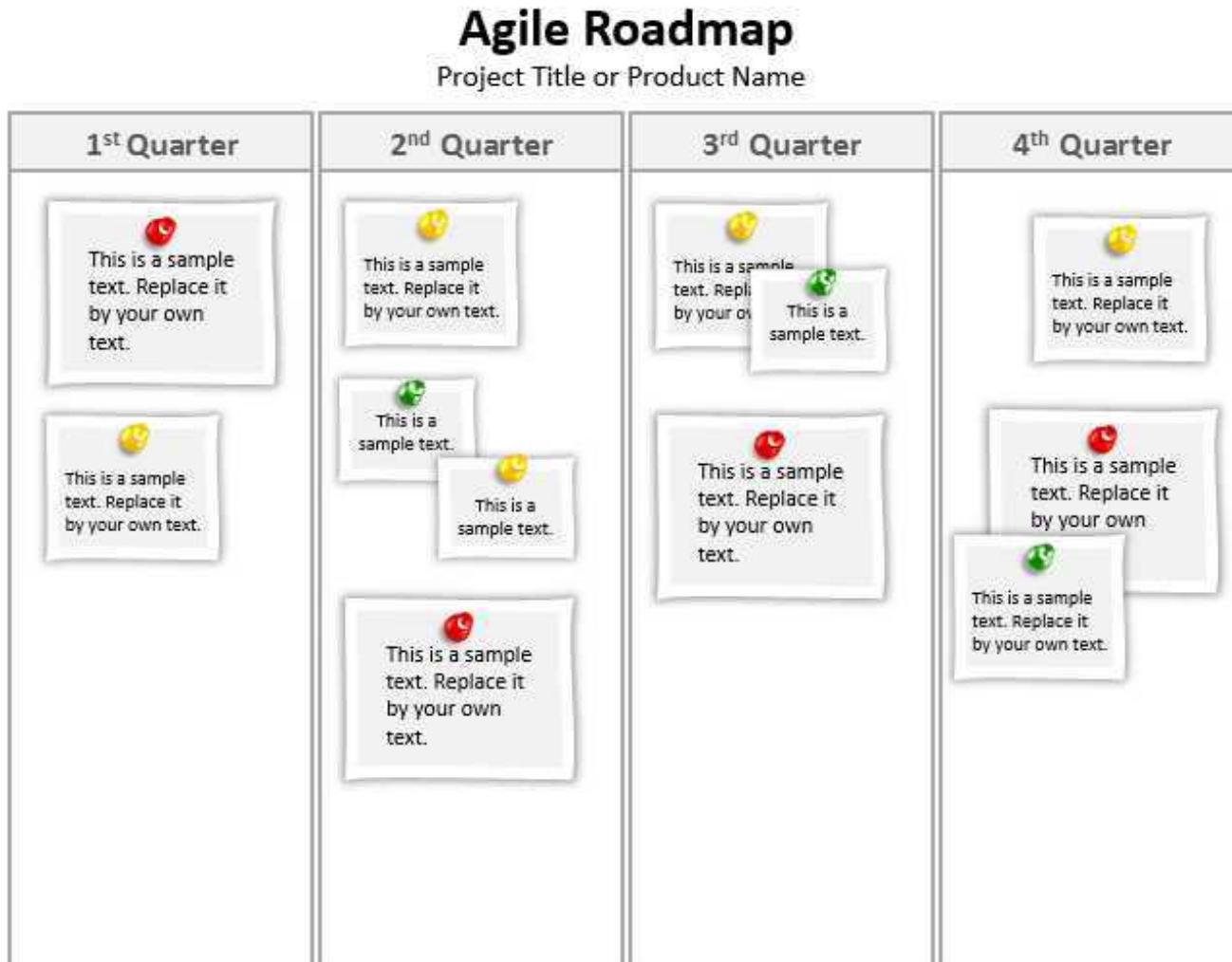
Planification Agile Roadmap



- On pose des jalons intermédiaires
- Les dates à l'intérieur ne sont pas nécessairement fixes ; ce sont les facteurs externes qui vont influer sur la durée des releases.
- La visibilité à moyen terme est faible, les fonctionnalités sont donc positionnées à titre indicatif dans la roadmap, tout changement pouvant intervenir dans l'ordre de priorité des fonctionnalités du PB, en fonction du client et des résultats des premières itérations

Planification Agile

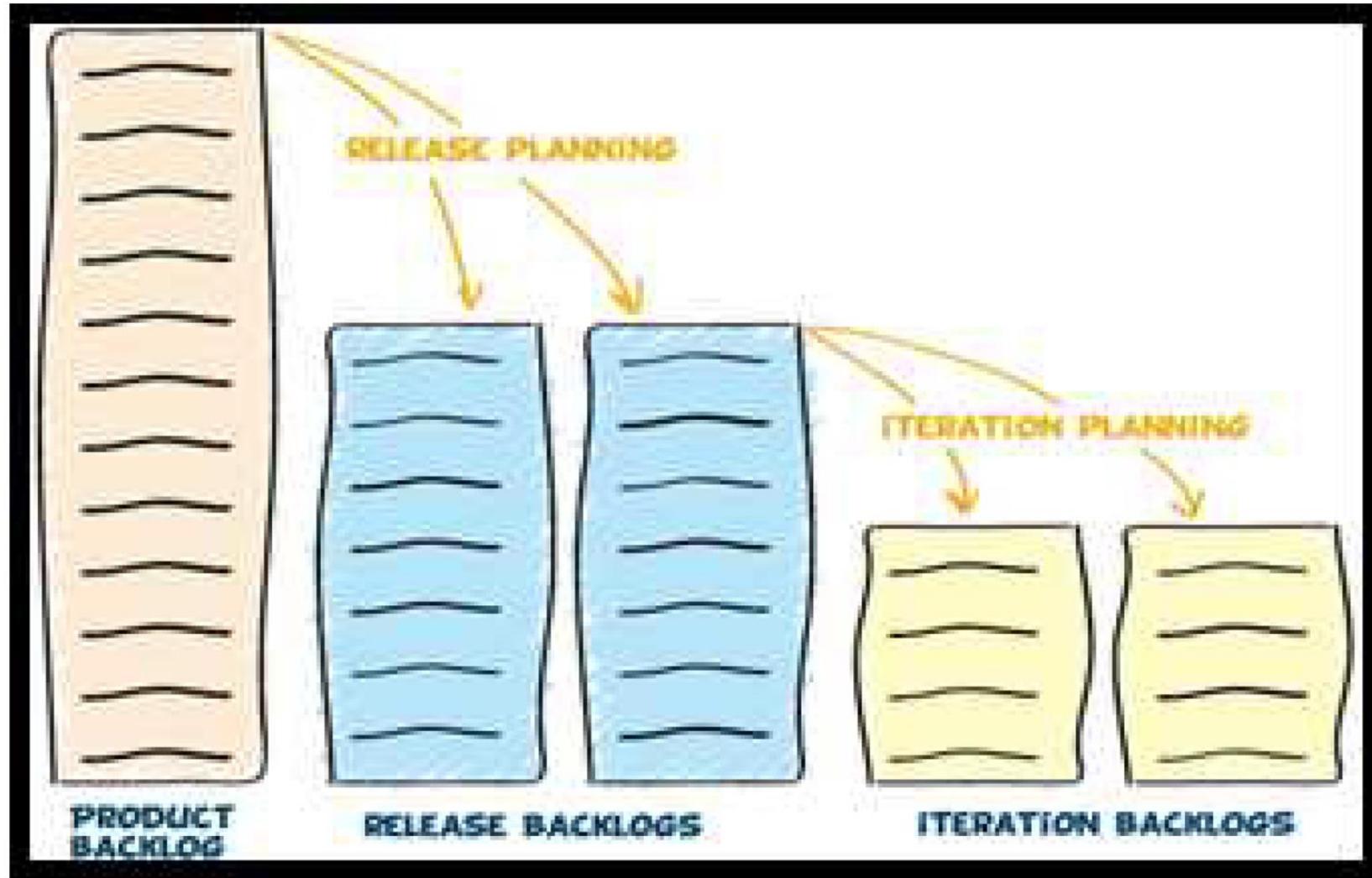
Roadmap (trimestrielle)



Planification Agile Roadmap (mensuelle)



Planification Agile Process depuis le PB



Planification Agile

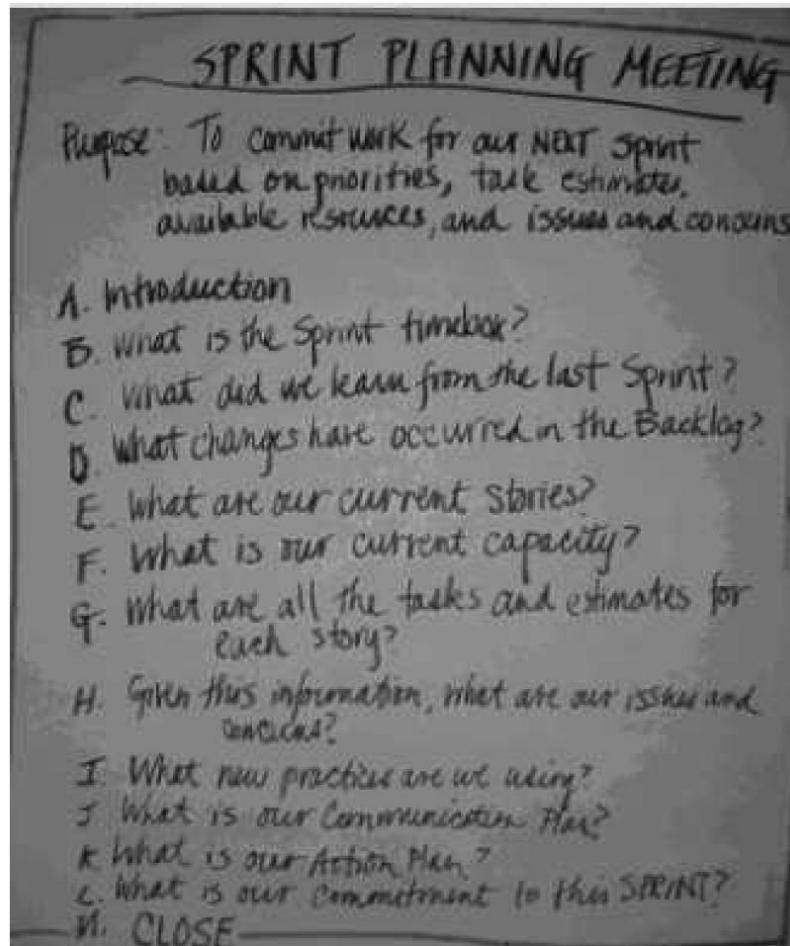
Plan de sprint (itération)

- Définir les stories à réaliser dans l'itération.
- Détailler les travaux à réaliser pour une story, recenser les activités et estimer le temps nécessaire à leur réalisation en heures (ideal hours) ou en points.
- Toutes les activités sont concernées : analyse, conception, développement, test, documentation... indispensables à l'implémentation complète d'une fonctionnalité, potentiellement livrable (potentially shippable) à la fin de l'itération. Une fonctionnalité n'est considérée comme achevée (done) que lorsqu'elle a été validée par le client et documentée. On ne différencie pas ces différentes disciplines, intégrées, par essence, dans une activité.

Planification Agile

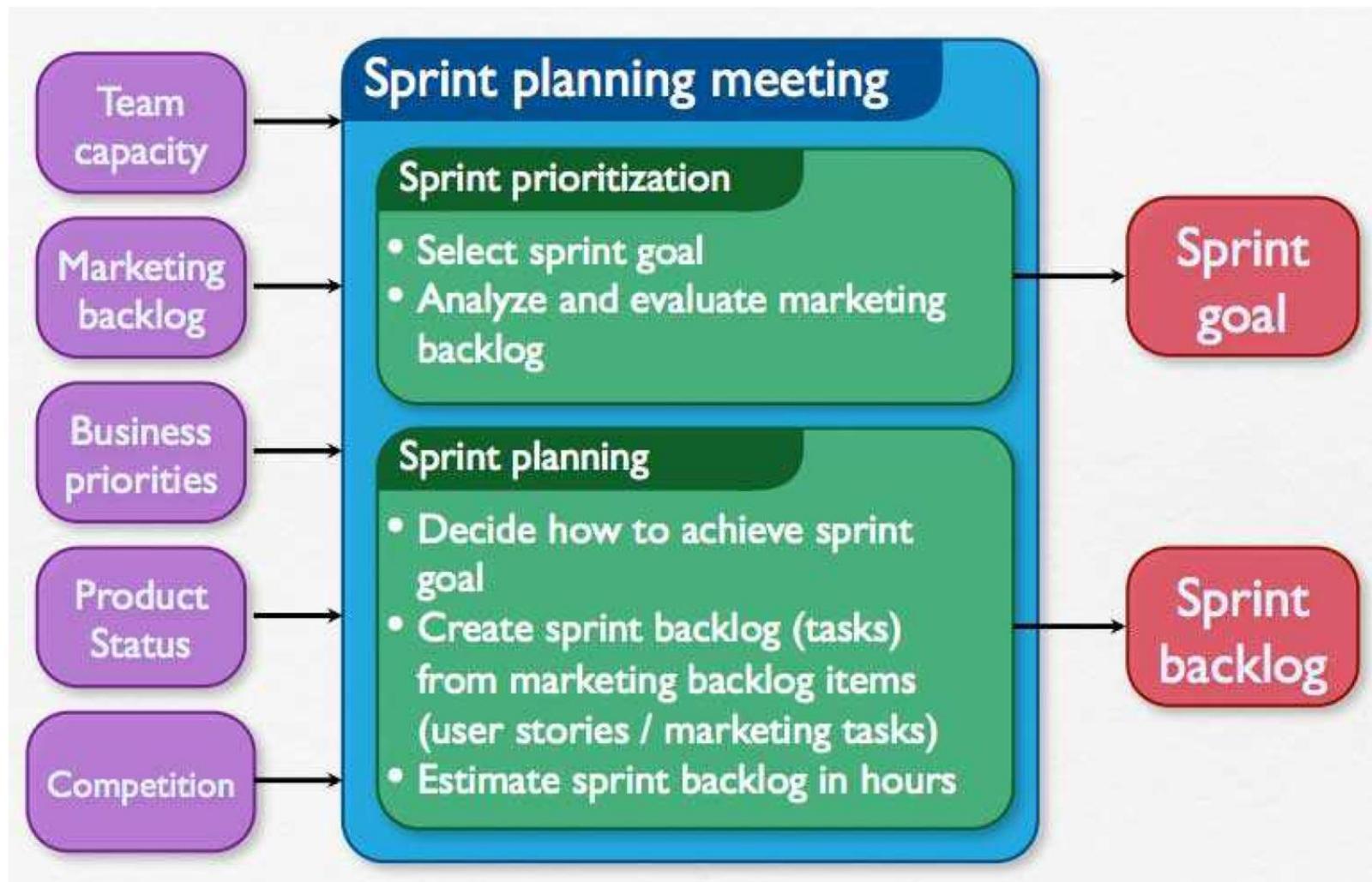
Plan de sprint

- Réunion de planification de sprint



Planification Agile

Plan de sprint (3)



Planification Agile

Daily stand-up meeting

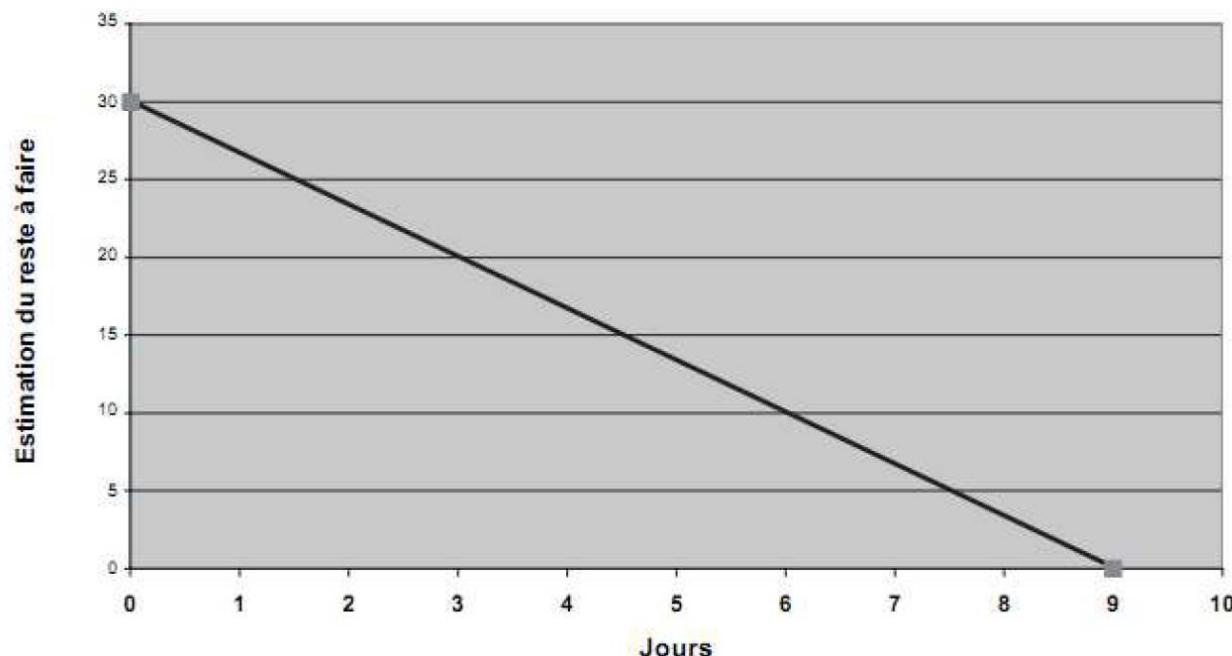


- Cycle quotidien = réunion pour suivre l'avancement de l'itération en cours (15mn maxi.)
- 3 questions systématiques auxquelles doit répondre chaque membre de l'équipe :
 - - ce qui a été fait la veille ?
 - - ce qui va être fait le jour même ?
 - - les éventuels blocages rencontrés ?
- **Intérêt** : pouvoir réajuster très rapidement le planning, l'organisation ou la répartition des tâches pour la journée à venir, et prendre des décisions tactiques, sans attendre la fin de l'itération.

Planification Agile

Sprint Burndown chart

- Planning de réalisation des tâches.
- Se présente sous forme d'une courbe idéale de réalisation du travail restant à faire (qui normalement décroît pour arriver à zéro en fin d'itération).



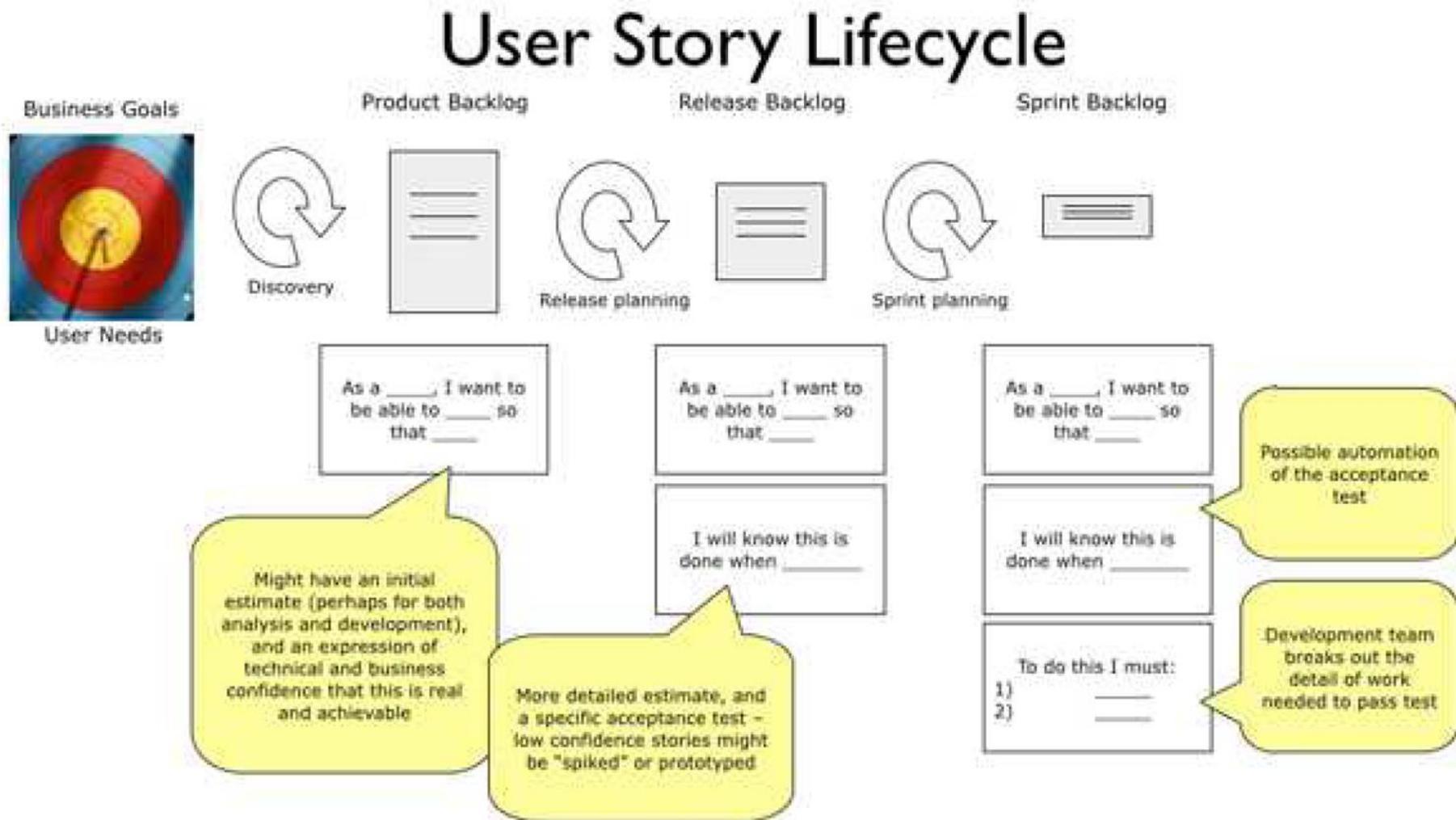
Synthèse des étapes de la planification



Démarche prédictive	Démarche agile
Étape 1 : calculer le délai	Étape 1 : estimation relative des fonctionnalités
Étape 2 : calculer le coût	Étape 2 : affectation des fonctionnalités aux différentes releases (roadmap)
Étape 3 : recenser les activités	Étape 3 (release) : clarification des fonctionnalités (user stories) et affectation aux différentes itérations
Étape 4 : calculer la durée des activités	Étape 4 (itération) : recensement des tâches pour chaque story traitée
Étape 5 : ordonner les activités	Étape 5 (réunion quotidienne) : réajustement du planning, répartition des tâches
Étape 6 : établir le planning	
Étape 7 : ajuster le planning	

Synthèse

Cycle de vie des user stories



Suivi et pilotage du projet



Suivi et pilotage du projet

- Pouvoir identifier où l'on en est (constat) et où l'on en sera réellement (projection, par anticipation, sur la fin du projet)
-
- Nécessité d'avoir un suivi rigoureux (indicateurs du tableau de bord pertinents).
- Démarche classique : suivre et analyser les écarts par rapport à un plan préétabli ;
- **Démarche agile** : s'adapter en permanence pour mieux progresser vers l'objectif que l'on s'est fixé.
- **Indicateurs** : performance, qualité, risque.

Indicateurs de suivi et pilotage

Performance

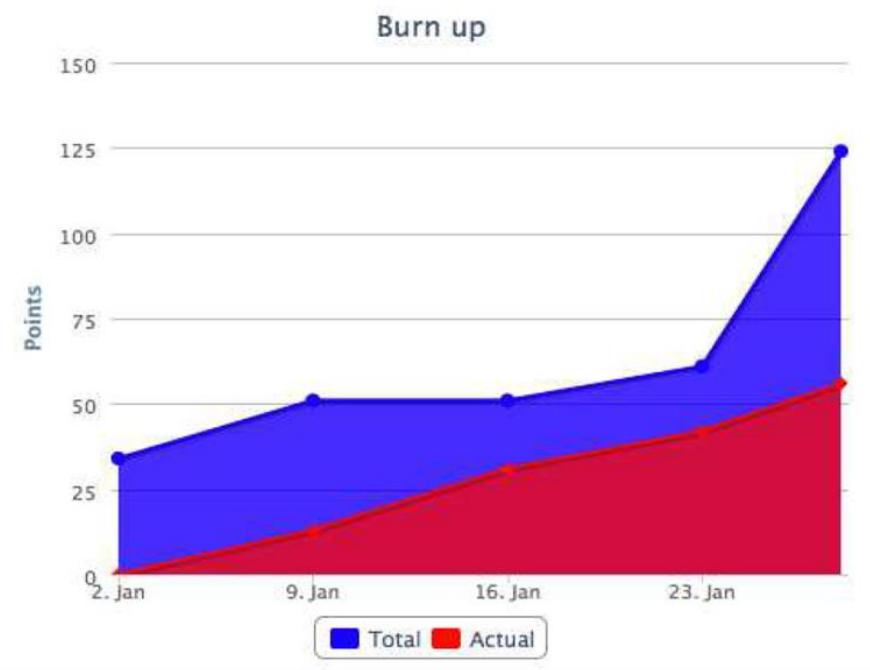
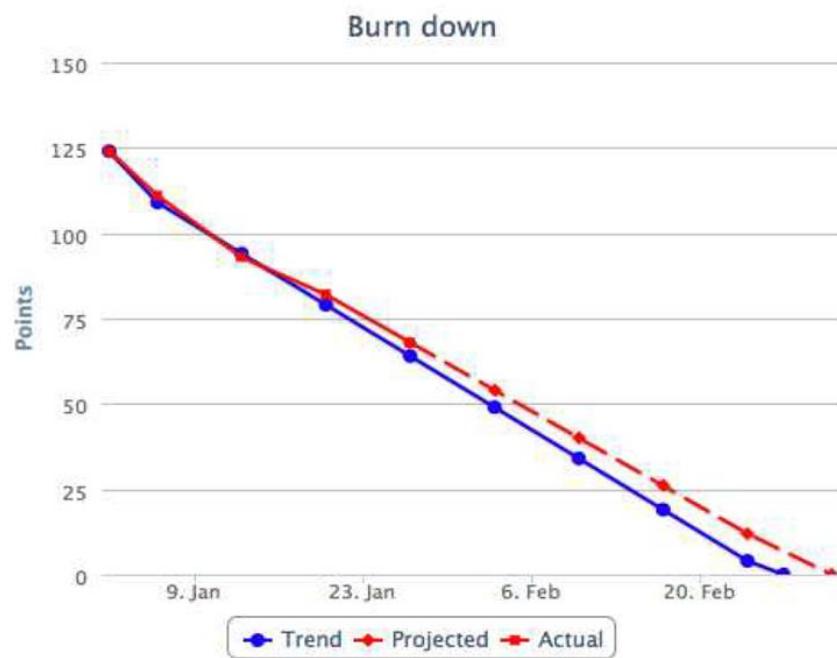


- Performance d'un projet à un instant t = la part de l'ensemble des travaux qui a été réellement réalisée : « Qu'avons-nous produit, à ce jour ? »
- Démarche classique : technique de la valeur acquise (EVM : Earned Value Management).
- **Démarche agile** : la performance ou l'avancement technique se mesure, non pas à l'avancement des activités, mais au **nombre de fonctionnalités développées et validées par le client**.

Indicateurs de suivi et pilotage

Performance (2)

- Burndown chart : le travail restant à faire qui décroît avec l'avancement du projet.
- Burnup chart : matérialise le nombre de fonctionnalités implémentées au fur et à mesure.



Indicateurs de suivi et pilotage

Performance (3)



- Mise à jour quotidienne par la saisie du travail restant à faire. Chaque collaborateur évalue la quantité de travail qu'il considère devoir consommer pour la réalisation des activités dont il a la responsabilité ; certaines d'entre elles peuvent se révéler plus coûteuses que prévu, d'autres suivent l'estimation initiale.
- Mesurer l'écart entre la courbe idéale du départ et la réalité du projet.

Indicateurs de suivi et pilotage

Qualité



- Les activités réalisées ou les fonctionnalités développées doivent remplir les critères d'évaluation et satisfaire le client !.
- La non-qualité peut se résumer et prend des formes différentes :
 - non-conformités,
 - anomalies,
 - bogues,
 - sur-qualité,
 - gaspillage...

Indicateurs de suivi et pilotage

Qualité - contrôle



- Contrôle qualité : effort de détection des défauts
- Peut être déclenché après les développements, considérant que, pour effectuer ces contrôles, les activités de codage doivent être achevées. Mais il peut être également mené dès les premiers développements afin de dépister et de corriger au plus tôt d'éventuelles anomalies.
- Moyens de prévenir la non-qualité : test, les revues ou les audits.

Indicateurs de suivi et pilotage

Qualité - types de test



- **Fonctionnel** : Test portant sur les fonctionnalités du système ; scénarios de test ou test case rédigés à partir des UC ou des user stories. (point de vue utilisateur)
- **Technique** : Test portant sur la satisfaction des exigences non fonctionnelles : temps de réponse, montée en charge, consommation réseau, sécurité,...
- **Interface** : Test portant sur la couche présentation, IHM, comportement des objets d'un écran (champs, menus, navigation...). + Test portant sur les interfaces entre systèmes : format des données échangées, communication en temps réel ou en batch...

Dette Technique

- 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

Se préoccuper de la dette technique

- Il est important que le Product Owner veille à ce que la dette technique de son produit soit maîtrisée. (un code propre et maintenable n'est pas une perte de temps).
- La dette technique concerne :
 - - la maintenance
 - - l'évolutivité d'un produit
 - - la fiabilité

Catégoriser la dette technique

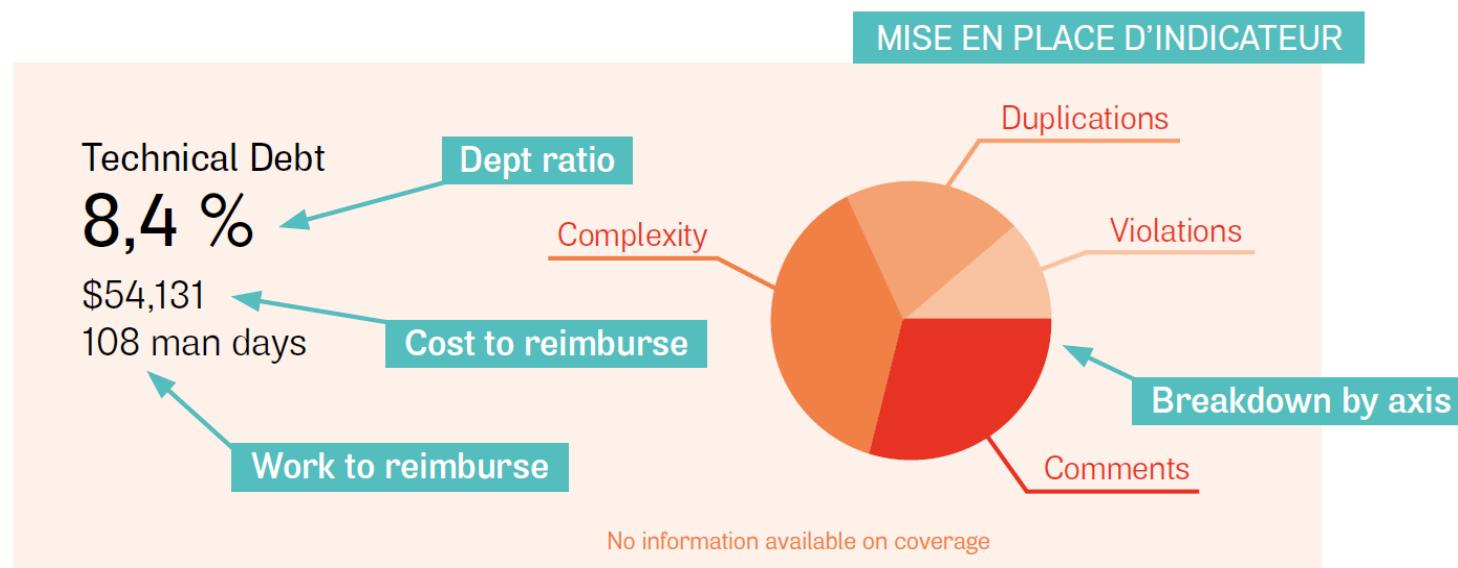
Le **Technical Debt Quadrant** (Martin Fowler) catégorise la dette technique en quatre types bien distincts :

TECHNICAL DEBT QUADRANT			
VOLONTAIRE	EXCESSIVE ET IMPRUDENTE	MODÉRÉE ET PRUDENTE	INVOLONTAIRE
	<p>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.</p> <p>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.</p>	<p>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.</p> <p>Ex : une partie de code rarement modifiée.</p>	<p>Dette souvent due à des équipes ignorant les pratiques de conception et d’architecture les plus basiques.</p> <p>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.</p>
			<p>Dette technique existant dans les équipes même 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.</p>

Mesurer la dette technique

Mettre en place des outils pour mesurer la dette technique :

SonarQube. 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 !



Réduire la dette technique (plan d'action)

• Faire la revue des indicateurs à chaque rétrospective, la dette technique sera ainsi un sujet régulièrement étudié par l'équipe.

● **Indexer 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.**

● Organiser 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.**

Encourager 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 !

Indicateurs de suivi et pilotage

Risques



- Objectif : anticiper sur le déroulement du projet et de prendre des décisions sur les actions à mener.
- Imagine les scénarios catastrophes, en identifiant les risques et les menaces qui pourraient compromettre le succès du projet.
- Activités :
 - - Recenser et caractériser les risques .
 - - Analyser et valoriser les risques.
 - - Formaliser les risques.
 - - Planifier les réponses aux risques.
 - - Surveiller et maîtriser les risques.

Indicateurs de suivi et pilotage

Risques - actions



- L'évitement : compétence externe ou suppression de l'exigence.
- Le transfert vers un tiers (sous-traitant).
- La réduction.
- L'acceptation.
- La surveillance

Présentation des indicateurs de suivi et pilotage

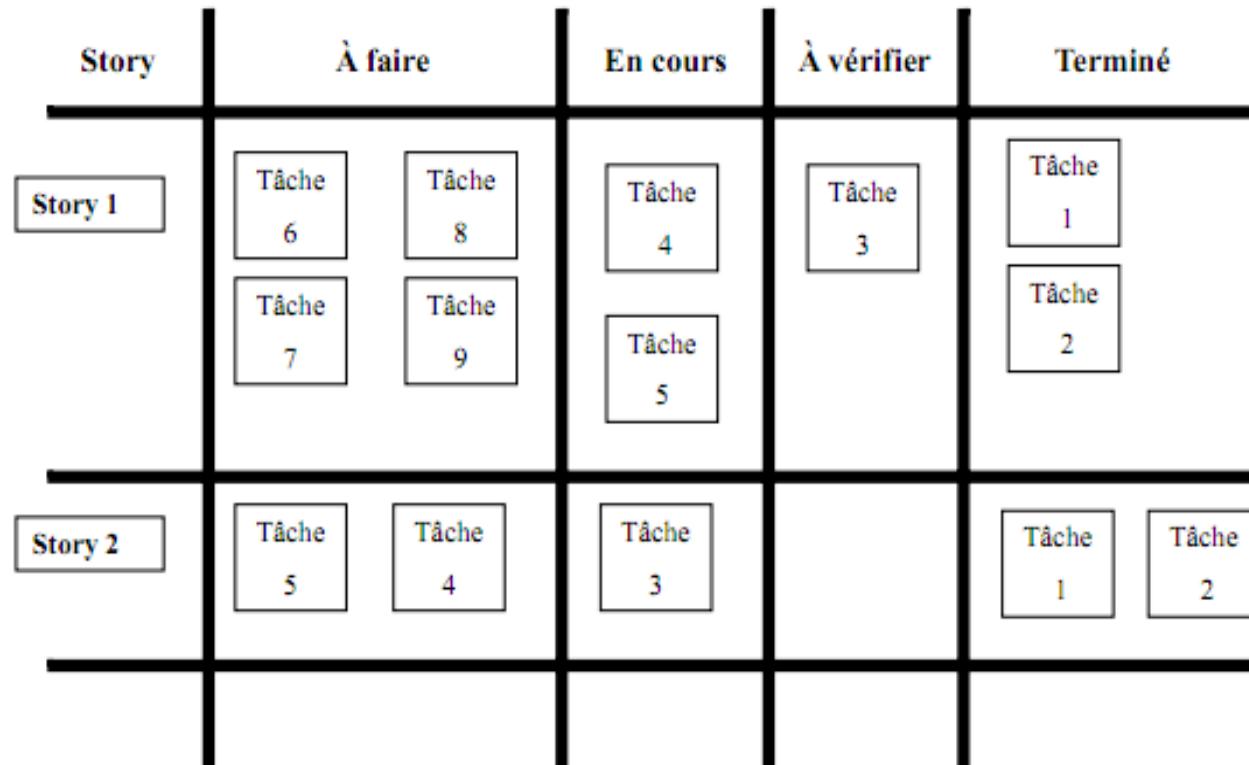


- **Reporting transparent (ne pas dissimuler au PO).**
- **Indicateurs à présenter :**
 - l'avancement technique, exprimé en pourcentage ou en nombre de fonctionnalités développées ;
 - l'écart en coût et en délai, au moment du point d'avancement et à l'achèvement du projet ;
 - l'estimation du reste à faire, en nombre de fonctionnalités, en jour/homme ou en points ;
 - le nombre de tests rédigés, passés ; le pourcentage de tests passés avec succès ;
 - le nombre d'anomalies, leur répartition par type ;
 - l'état des risques ;
 - les faits marquants depuis le dernier rapport, les décisions prises, les actions mises en œuvre

Présentation des indicateurs

Tasks board

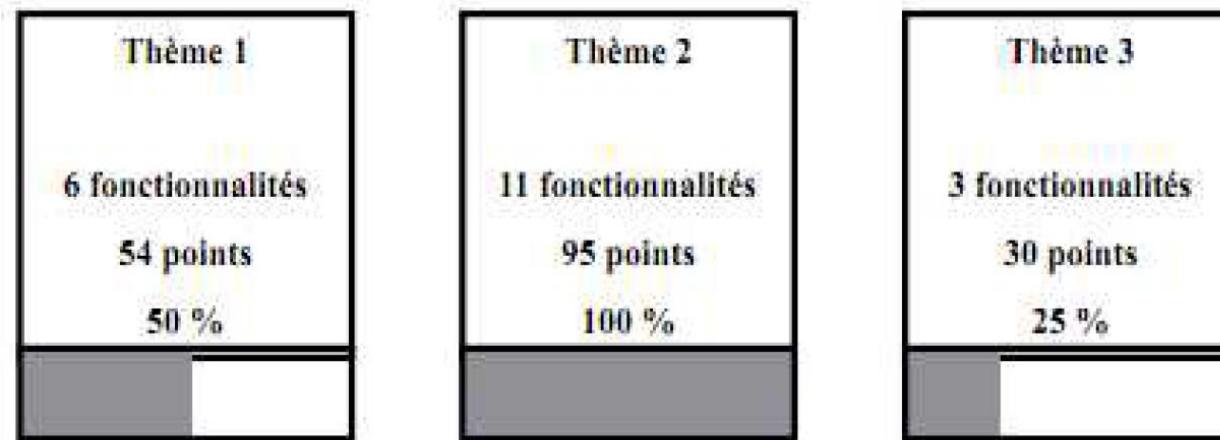
- Formalise : tableau+post-it ou logiciel dédié
- Chaque membre de l'équipe prend une carte dans la 2ème colonne, y inscrit ses initiales et la déplace, au fur et à mesure que le travail s'achève... Les tâches ne sont pas affectées aux collaborateurs mais choisies par eux sur la base du volontariat.



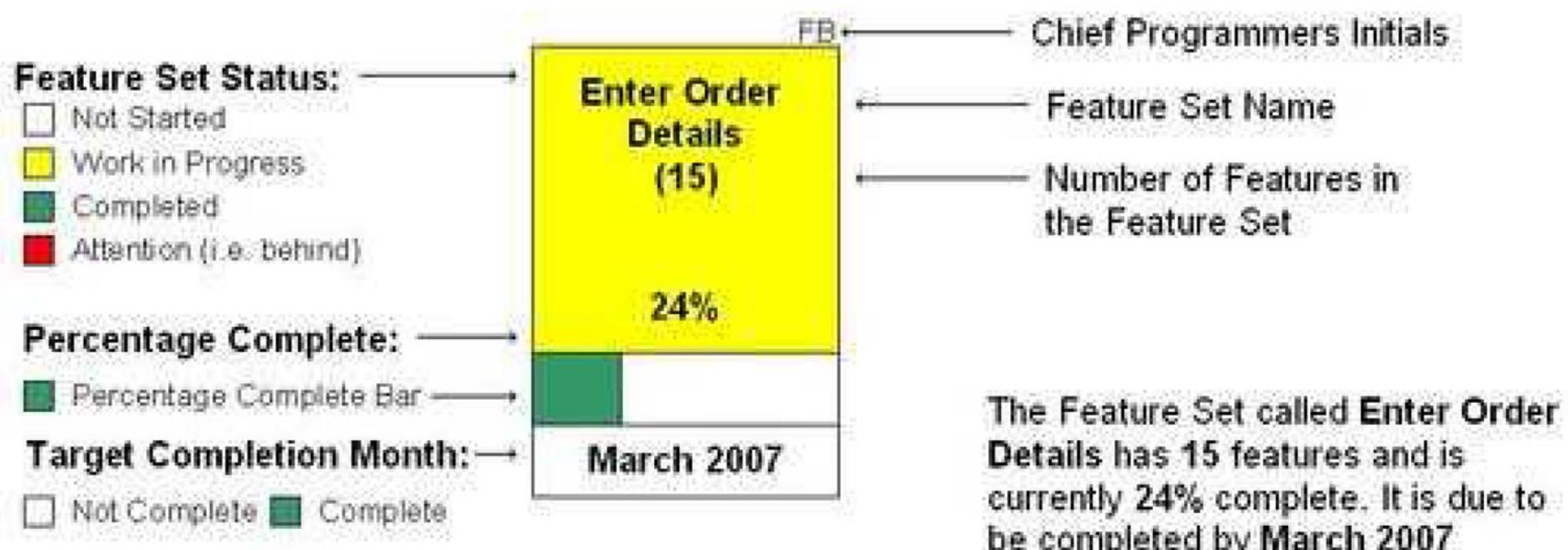
Présentation des indicateurs

Parking Lot

- Chaque rectangle représente un thème ou un sous-ensemble de fonctionnalités dans une release : il indique le nom du thème, le nombre de fonctionnalités, le nombre de points estimé pour ces fonctionnalités et le pourcentage de fonctionnalités achevées dans une barre de progression.
- Particulièrement adapté sur des gros projets avec un nombre important de fonctionnalités que l'on regroupe par thèmes.



Présentation des indicateurs Parking Lot (2)



Revue de Sprint (Sprint Review)



- Inspecter l'incrément produit au cours du sprint écoulé, faire un point sur l'avancement de la Release et adapter au besoin le plan et le Product Backlog.
Durée : 4h pour un sprint d'un mois.
- L'équipe de développement présente à tout acteur projet intéressé (à minima le Product Owner idéalement accompagné d'utilisateurs finaux) les nouvelles fonctionnalités développées au cours du sprint.
- Le Product Owner donne un feedback à l'équipe de développement, il accepte ou refuse les fonctionnalités présentées.

Revue de Sprint (2)

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

Sprint Retrospective

- Cette réunion est généralement animée par le Scrum Master qui s'adresse à son équipe.
(Outil d'animation - cartes : l'art de la rétrospective)
- Elle a pour but d'améliorer continuellement le processus de développement de l'équipe en mettant les idées de chacun à contribution.
- Durée : 3h pour un sprint d'un mois
- Approche : identifier et pondérer les éléments positifs (éléments à cultiver ou source de motivation) du sprint écoulé, puis les éléments à améliorer, puis de définir un plan d'action d'amélioration (en commençant par améliorer les éléments dont la pondération est la plus forte).

Sprint Retrospective (2)

- Commencer par les points positifs peut être vital lorsque le sprint a été éprouvant. L'équipe aura peut être besoin de se nourrir des éléments positifs avant de s'attaquer aux éléments à améliorer.
- Tous les domaines peuvent être abordés en rétrospective : **humain, organisationnel, pratiques, processus, outillage, qualité de vie au travail, conflits, interactions avec le métier**. Le compte rendu de la dernière rétrospective, les graphiques d'avancement de sprint et release, les événements du sprint, les feedbacks de la revue de sprint sont autant d'éléments qui alimentent les conversations.

Sprint Retrospective (3)

Exemples de thèmes



- Acteurs externes
- Équipe
- Client
- Outils
- Méthode
- Web
- Tests
- ...

Sprint Retrospective

Agenda / Pratique

- Revue des actions prises à la rétrospective précédente
- Collecte des données: atelier, innovation game
- Consolidation et priorisation des sujets
- Établir le plan d'actions
- Clôture / ROTI (Return Of Time Invested)

•Pratique :

Comment garder les équipes actives et intéressées ?

Comment éviter la monotonie des rétros toujours les mêmes ?

Comment trouver une idée différente et originale à chaque itération ?

=> **innover avec des games** : Speed boats, Turn The Tables, Jeopardy,

Sprint Retrospective

Speed Boat



- Cet innovation game peut s'avérer utile dans différents contextes, il va permettre à l'équipe de réfléchir à sa vision et ses objectifs, à ce qui les ralentis dans l'atteinte de ces derniers, les risques qu'ils peuvent rencontrer sur le chemin et les bonnes pratiques et les forces qui vont les aider à atteindre leur but.



Sprint Retrospective

Speed Boat (2)



Voici ce que symbolise chacun des éléments :

- Le bateau : l'équipe
- L'île : les objectifs à atteindre
- Les vents : nos forces, ce qui nous fait avancer
- Les ancles : ce qui nous ralentit, les freins
- Le soleil : qui je remercie, ce que j'ai aimé
- Le rocher : les risques futurs (obstacles à venir)

Sprint Retrospective

Speed Boat (3)



Déroulement :

- Chaque personne dispose de 5 minutes pour préparer individuellement ses post-its pour les différents éléments du dessin.
- Chacun des membres de l'équipe vient coller ses post-its sur le dessin en expliquant rapidement chaque point. À cette étape, il vaut mieux déjà commencer à regrouper les post-its qui parlent du même sujet.
- L'équipe vote pour le ou les sujets qui lui semble(nt) les plus prioritaires à traiter (dot voting). Limitez votre nombre de sujets, il vaut mieux s'engager sur un seul sujet sur lequel on arrivera à mettre en place les actions rapidement plutôt qu'une multitude de sujets partiellement réalisés. L'amélioration continue doit s'effectuer par petits pas mesurables.
- Sur chacun des sujets l'équipe élabore un plan d'actions qui seront mises en place dès le sprint suivant.

Sprint Retrospective

Turn the tables



- Etapes :

- 1- Mise en place
- 2- Récolter les données
- 3- Générer les idées
- 4- Décider quoi faire
- 5- Clore la rétrospective

Le jeu se concentrera autour des phases 2 (Récolter les données) et 3 (Générer les idées) avec une idée innovante. Vous pourrez choisir les activités des autres étapes parmi les nombreuses activités classiques.

- Matériel : des cartes (autocollantes ou pas) vertes et rouges, un tableau blanc.

Temps nécessaire : environ 1 heure.

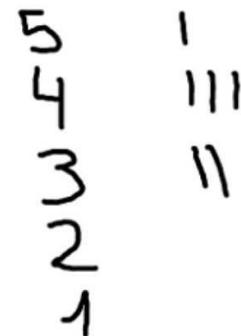
Sprint Retrospective

Turn the tables (2)

1 - Mise en place

- Si première rétrospective => rappel des principes
- « Check-in » de l'équipe (pour avoir l'état d'esprit) :
On peut demander à chaque collaborateur comment il a vécu le sprint courant, sur une échelle de 1 (mal) à 5 (très bien). Vous pouvez ensuite dessiner les résultats sur le tableau blanc

HOW WAS THE
SPRINT?



Sprint Retrospective

Turn the tables (3)

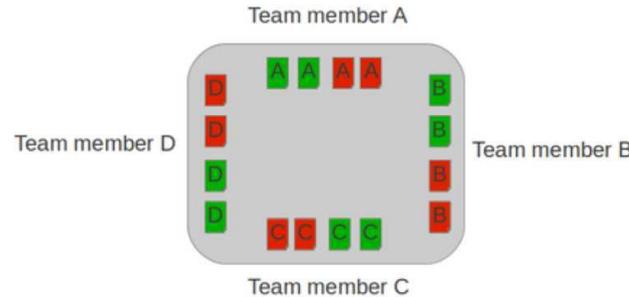


2 - Récolter les données

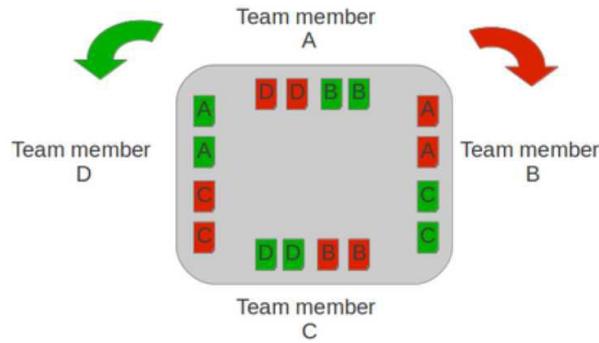
- Demandez à chaque participant d'écrire au maximum sur deux cartes rouges et deux cartes vertes les choses qui se sont bien déroulées pendant le sprint et les choses qui pourraient être améliorées. Demandez aux participants de n'écrire qu'un mot ou deux pour décrire au mieux leur idée (Il est important de respecter ces consignes).
- Si on vous demande pourquoi, ou si les participants vous disent que c'est contraignant, demandez leur de jouer le jeu et de vous faire confiance : ils comprendront très vite pourquoi.
- Si vous avez le matériel nécessaire, il est préférable d'écrire les choses positives sur les cartes vertes et les choses à améliorer sur les cartes rouges. Utilisez des marqueurs et écrivez assez gros pour que tout le monde dans la pièce puisse lire les cartes.

Sprint Retrospective

Turn the tables (4)



Une fois que tout le monde a fini, demandez aux participants de donner leurs deux cartes rouges à la personne située sur leur gauche, et leurs deux cartes vertes à la personne située sur leur droite.



Sprint Retrospective

Turn the tables (5)

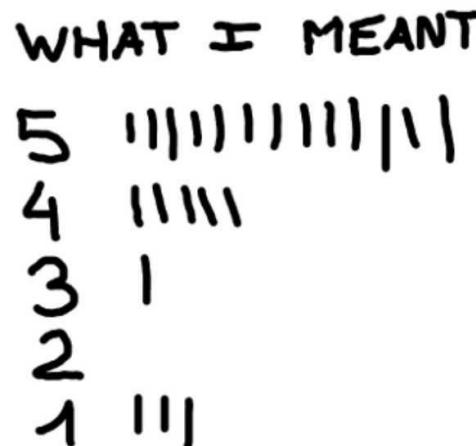
- Pourquoi il était important de n'écrire qu'un seul mot sur la carte : parce que c'est quelqu'un d'autre qui va présenter la carte à votre place !
- Demandez aux participants de présenter brièvement la carte qu'ils ont dans les mains (celle de leur co-équipier) en essayant de se mettre dans la peau de leur collaborateur.
- Cet exercice a plusieurs bénéfices. Tout d'abord, c'est amusant. Ensuite, il est intéressant de deviner ce que les autres veulent dire et d'imaginer comment ils ont vécu leur sprint. Parfois, des personnes travaillent à côté l'un de l'autre mais ne se parlent pas beaucoup.
- En ce sens, cet exercice peut améliorer la cohésion de l'équipe et atténuer les malentendus.

Sprint Retrospective

Turn the tables (6)

Variante (récolte des données) :

- Après chaque présentation de la carte, demandez à l'autre (original de la carte) de noter la pertinence de l'explication sur une échelle de 1 ("ce n'est pas du tout ce que je voulais dire") à 5 ("Je n'aurais pas pu mieux l'expliquer").
- Dessinez les notes au tableau : c'est un bon indicateur de la cohésion et de la communication au sein de l'équipe.



Sprint Retrospective

Turn the tables (7)

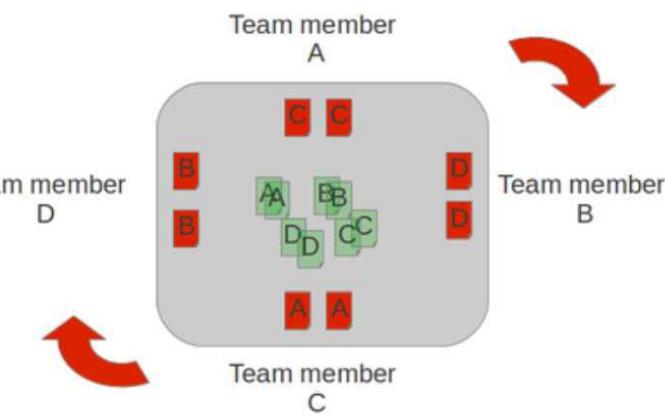
- Il peut arriver que les gens soient frustrés parce que l'explication ne comprend pas exactement à ce qu'ils voulaient dire. Au besoin : l'auteur de la carte peut compléter la description, mais **assurez vous que ce soit bref**. Cette étape se termine quand toutes les cartes ont été présentées.
- Si vous le voulez, vous pouvez demander aux participants de **commencer par les cartes vertes**, et tourner dans le sens des aiguilles d'une montre et recommencer avec les cartes rouges une fois que toutes les vertes ont été présentées.
- Vous pouvez aussi décider d'alterner une verte et une rouge par personne.
- C'est à vous de décider mais attention au temps ! En tant que Scrum Master, la gestion du temps est souvent votre responsabilité.

Sprint Retrospective

Turn the tables (8)

3 - Générer les idées

- Après que chaque carte ait été présentée, demandez aux participants de passer les cartes rouges qu'ils ont en main à la personne à leur gauche (les cartes rouges ne doivent pas revenir dans les mains de leur auteur). Laissez les cartes vertes au milieu de la table.
- Maintenant, les participants ont des cartes rouges (écrites par quelqu'un d'autre).



Sprint Retrospective

Turn the tables (9)

- Demandez alors à chaque personne de suggérer une action pour résoudre le problème décrit sur la carte rouge qu'il a en main. Tout le monde peut participer, mais il est à mon avis important que la première idée vienne de la personne qui tient la carte.
- En effet, il n'est pas rare que les meilleures solutions viennent de quelqu'un d'autre, quelqu'un avec un peu plus de recul et un point de vue différent. Il arrive parfois qu'on soit tellement concentré sur le problème qu'on passe à côté d'une solution évidente.
- Écrivez toutes les suggestions d'actions au tableau, en utilisant l'activité de votre choix ("short subjects" par exemple)

<u>Stop doing</u>	<u>Start doing</u>	<u>Keep doing</u>
- daily stand up minutes	- review of story with P.O. before starting to work on a new story - hire an agile coach?	- respect our team process - timeboxed daily stand up

Sprint Retrospective

Turn the tables (10)

4 – Décider quoi faire

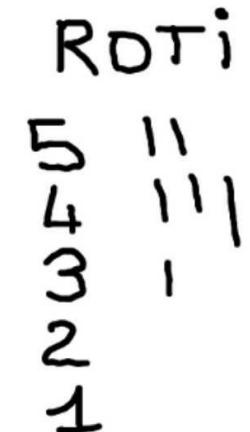
- Maintenant, vous avez tout un tas de propositions d'actions au tableau. Vous devriez en choisir 2 ou 3 à prendre dans le prochain sprint. Ne prenez pas trop d'actions, surtout si les actions demandent du temps. Ne prenez que des actions que votre équipe est capable d'adresser. Sinon, rétro après rétro, les membres de l'équipe vont commencer à se demander à quoi cela sert de prendre des actions si on ne les réalise jamais.
- Si vous avez beaucoup d'actions, utilisez n'importe quelle technique pour n'en choisir qu'un nombre limité.
- Par exemple, vous pouvez donner à chaque membre de l'équipe 3 votes et faire voter les gens pour leur action favorite. Prenez les 2 ou 3 actions qui ont remporté le plus de votes.
- Même si ça peut être frustrant au début, l'avantage du nombre limité de cartes par personne en début de rétro est qu'au final, vous ne devriez pas avoir trop d'actions proposées.

Sprint Retrospective

Turn the tables (11)

4 – Clôre la rétrospective

- Conclure la rétrospective en remerciant tout le monde pour sa participation. Vous pouvez également utiliser une activité de fin de rétrospective.
- Par exemple, vous pouvez utiliser le ROTI :
« comment les participants ont perçu le temps qu'ils ont investi en participant à cette réunion »
1 (“aucun bénéfice par rapport au temps investi”) à
5 (“Je n'aurais pas pu mieux utiliser mon temps”).



Sprint Retrospective

Turn the tables (12)

Turn the tables est recommandée dans les cas suivants

- une équipe pas trop grande (3 à 7p)
(afin de garder la rétro. courte)
- si vous observez des problèmes de communication dans l'équipe
- si il y a des tensions entre les membres de l'équipe
- si vous souhaitez réduire le nombre de posts-its
- quand les membres de l'équipe ne se connaissent pas beaucoup
(ou en début de projet, après 2 ou 3 itérations)

Sprint Retrospective

Joepardy



- Célèbre jeu télévisé où pour gagner, il faut trouver la bonne question à une réponse donnée. C'est sur ce principe que ce format de rétrospective est construit :
- 1 - Chaque équipier réfléchit individuellement à une ou plusieurs questions (qu'il garde secrètement pour lui) et note la réponse sur un post-it (toujours suivant le principe : une idée = un post-it),
- 2 - Toutes les réponses sont affichées au tableau pour former un tableau de jeu,
- 3 - Chaque équipier est invité à imaginer les questions aux réponses qui ne sont pas les siennes,
- 4 - Toutes les questions sont notées à côté des réponses (à ce moment-là, généralement, l'auteur de la réponse a une réaction),

Sprint Retrospective

Jeopardy (2)

5 - Quand les équipes sont à court de questions et que toutes les réponses ont été couvertes, les auteurs révèlent leurs questions originales,

6 - C'est le moment des réactions : surprises, découvertes, hallucinations, éclats de rire, incompréhensions, quiproquos, parfois déceptions (ça arrive),

7 - Les équipes votent pour les questions qu'ils estiment les plus importantes (par exemple en utilisant la technique du Dot voting),

8 - L'équipe décide à minima d'une action réalisable d'ici à la prochaine rétrospective.

• Ce format qui permet à chacun d'exprimer un sujet, de susciter des réactions, de trouver des actions (elles se trouvent parfois dans la question), de confronter les points de vue et d'apprécier les divergences sur des sujets de la vie de l'équipe.

Sprint Retrospective

Joepardy (3)

Colonne en cours et fini	Parallélisation
De quoi peut-on se passer ? Qu'est-ce qui a disparu mystérieusement ?	Qu'est-ce qui manque cruellement ? Qu'est-ce qui pourrait améliorer les performances ? Qu'est-ce qui a bien fonctionné ?
Absence de Pierre	Perturbations
Qui était absent ? Qu'est-ce qui a accélérer le sprint ? Qu'est-ce qui n'était pas prévu dans le sprint ?	Qu'est-ce qui ne manque pas cruellement ? Qu'est-ce qui n'a pas perturbé le burndown chart ? Qu'a-t-on mal géré en tant qu'équipe ?
Décompte du burndown de releaser	Niveau hétérogène des compétences
Qu'est-ce qui est optimal ? Qu'est-ce qui est rouge et qui descend ? Qu'est-ce qui n'existe pas ? Dans quoi ce sprint n'est pas pris en compte ?	Qu'est-ce qui ralentit certains développeurs ? Que doit-on prendre en compte lors du Spring planning ? Qu'est-ce qui rend le planning de sprint plus difficile ?

gestion des ressources humaines



Scrum Master (Manager agile)



- Le chef de projet est devenu un facilitateur, qui a pour responsabilité de créer les meilleures conditions, humaines et environnementales, favorisant la collaboration et facilitant le succès du projet.

• Responsabilités :

- Constituer l'équipe
- Définir les rôles et responsabilités
- Déterminer la composition de l'équipe
- Contracter les ressources
- Animer l'équipe
- Développer la collaboration
- Créer un environnement de travail efficace
- Gérer les sous-traitants

Créer un environnement de travail efficace



- Réunir l'équipe dans une même salle (open space?).
- Disposition pour faciliter le travail en binôme.
- Panneau d'affichage des résultats.
- Prévoir un poste pour le client (qui passe régulièrement du temps avec l'équipe, pour décrire les fonctionnalités attendues ou pour tester les derniers développements)

Gérer des équipes multiples ou distantes



- Maintenir la taille des équipes entre 3 et 9 personnes au maximum (afin de limiter les échanges).
- Tenir une réunion quotidienne pour chaque équipe, suivie d'une réunion avec un représentant de chaque équipe (scrum of scrum).
- Introduire des standards entre les équipes (codage, dépôt de sources, métriques identiques).
- Partager le Product Backlog.
- Utiliser des outils de communication dédiés pour le travail collaboratif et le partage des indicateurs de suivi.

Aspects contractuels

Aspect contractuel

Comment gérer un projet Scrum sur le plan contractuel, sachant que les concepts Agiles préfèrent la collaboration à la négociation de contrat ?

- De la part du soumissionnaire, un engagement sur les moyens (compétences, méthodes, pilotage, ...) pour assurer la qualité
 - Estimation du rapport qualité/prix plutôt que du prix le plus bas
 - Exemples de critères : niveau de qualité (méthodes, ...), fréquence des feedbacks, expériences des ressources, capacité à s'adapter, communication, maturité de la relation client
- Un budget et des dates de livraisons (durée du sprint) fixés par le client
- Un périmètre initial tracé de façon collaborative (backlog produit initialisé conjointement)

Scrum

Aspect contractuel (2)



- Contrat à engagement de moyens (~régie) : périmètre variable et coût fixe
 - Risques principaux : retard de livraison, dé-responsabilisation du soumissionnaire
- Co-sourcing
 - Équipe mixte, sur le site du client
 - Visibilité, transfert de compétences intrinsèque, risques partagés (des ressources des deux parties sont impliquées)

Scrum

Aspect contractuel (3)

Suite approches possibles pour l'agilité :

- **Forfait collaboratif**

- Le coût et le délai sont fixés par le client ▶ le périmètre est la variable d'ajustement
- Le soumissionnaire se positionne sur sa capacité et son savoir-faire (engagement sur des critères de qualité)
- Le contrat a donc bien un prix et une date fixés. La différence est dans la fréquence de livraison et dans l'ajustement du périmètre.
- Forte implication des parties

- **Forfait avec engagement mesurable**

- Définition de points de mesures concrets et représentatifs (par exemple : Définition d'unités d'œuvre, engagement sur un niveau de qualité, ...)

Mise en œuvre

- Le client donne le prix P et le délai D
- Le fournisseur et le client définissent ensemble la durée d'une itération I
Le tarif par itération est PxI/D
- A la fin de chaque itération, le client voit ce qui fonctionne et peut décider de :
 - changer de fournisseur, le client est payé : $nxPxI/D$ (plus un pourcentage lié à l'investissement)
 - changer de priorité dans les exigences.

Adopter une approche agile



Adopter une approche agile

- Dresser l'état des lieux
- Recenser les zones de dysfonctionnement
- Poser les bonnes questions
- Risques ou facteurs clés de réussite ?
- Fixer des objectifs réalistes
- Comment mesurer le succès de la démarche ?
- Quels sont les symptômes de l'échec de la démarche ?
- Initialiser la conduite du changement

Conclusion

- Déployer une nouvelle méthodologie doit être considéré comme un projet à part entière qui s'adossera à un ou deux projets de développement pilotes, retenus pour l'expérimentation.
- On accompagnera la démarche d'un plan de conduite de changement, pour sensibiliser les acteurs aux enjeux et aux résultats tangibles, pour les former, les convaincre et favoriser leur adhésion.

Plus d'informations sur <http://www.dawan.fr>

Contactez notre service commercial au
09.72.37.73.73 (prix d'un appel local)

