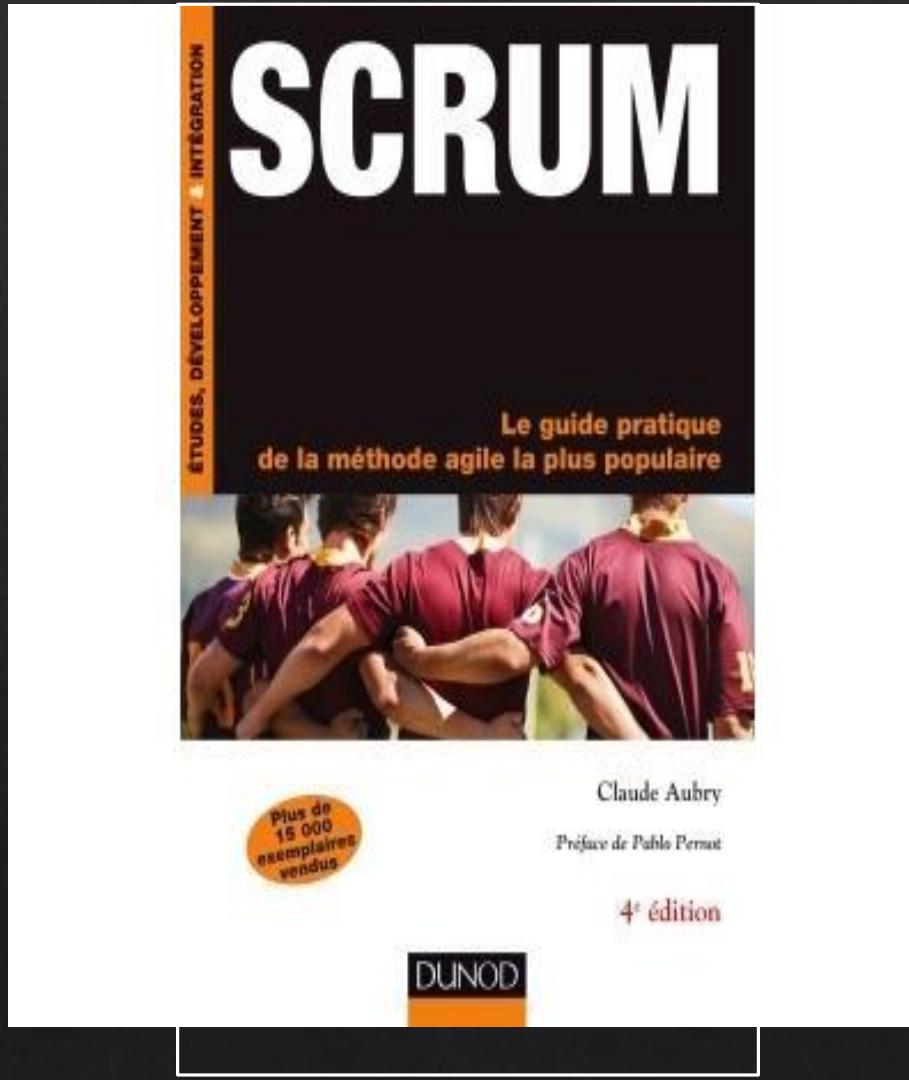
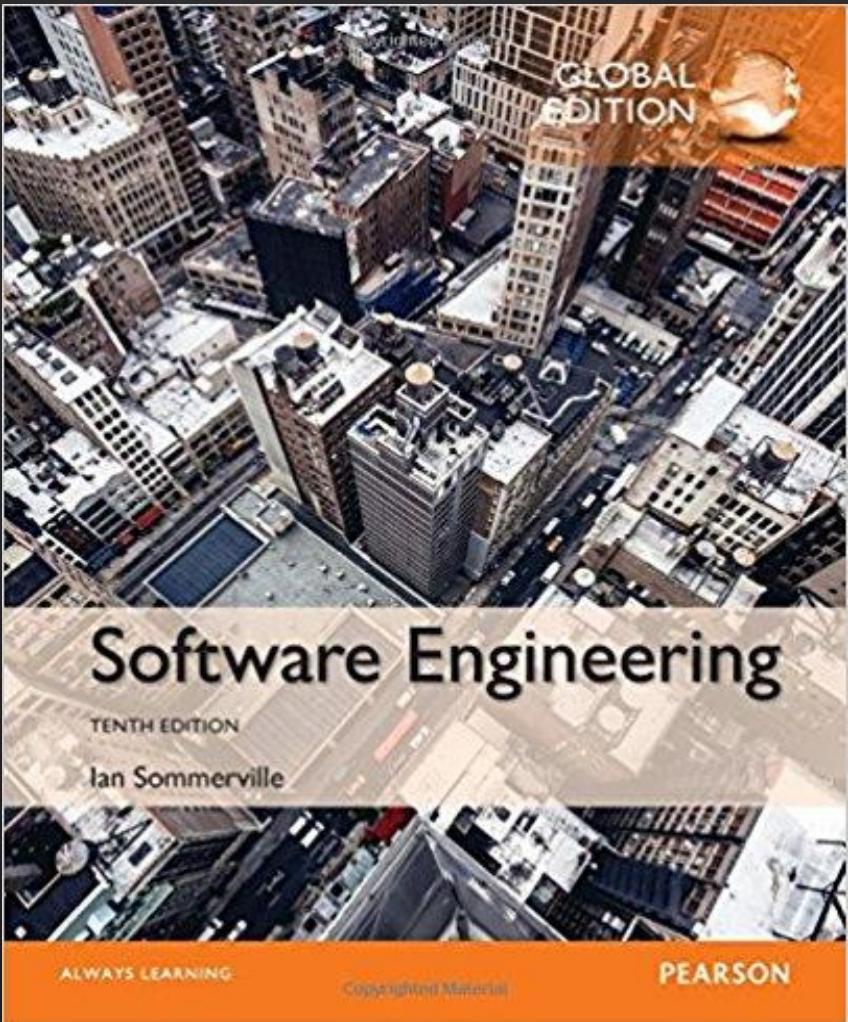


# Les processus agiles - SCRUM

# Sources



# Les processus dits agiles

# Le manifeste agile -Valeurs

- ❖ Les individus et leurs interactions plus que les processus et les outils
- ❖ Des logiciels opérationnels plus qu'une documentation exhaustive
- ❖ La collaboration avec les clients plus que la négociation contractuelle
- ❖ L'adaptation au changement plus que le suivi d'un plan

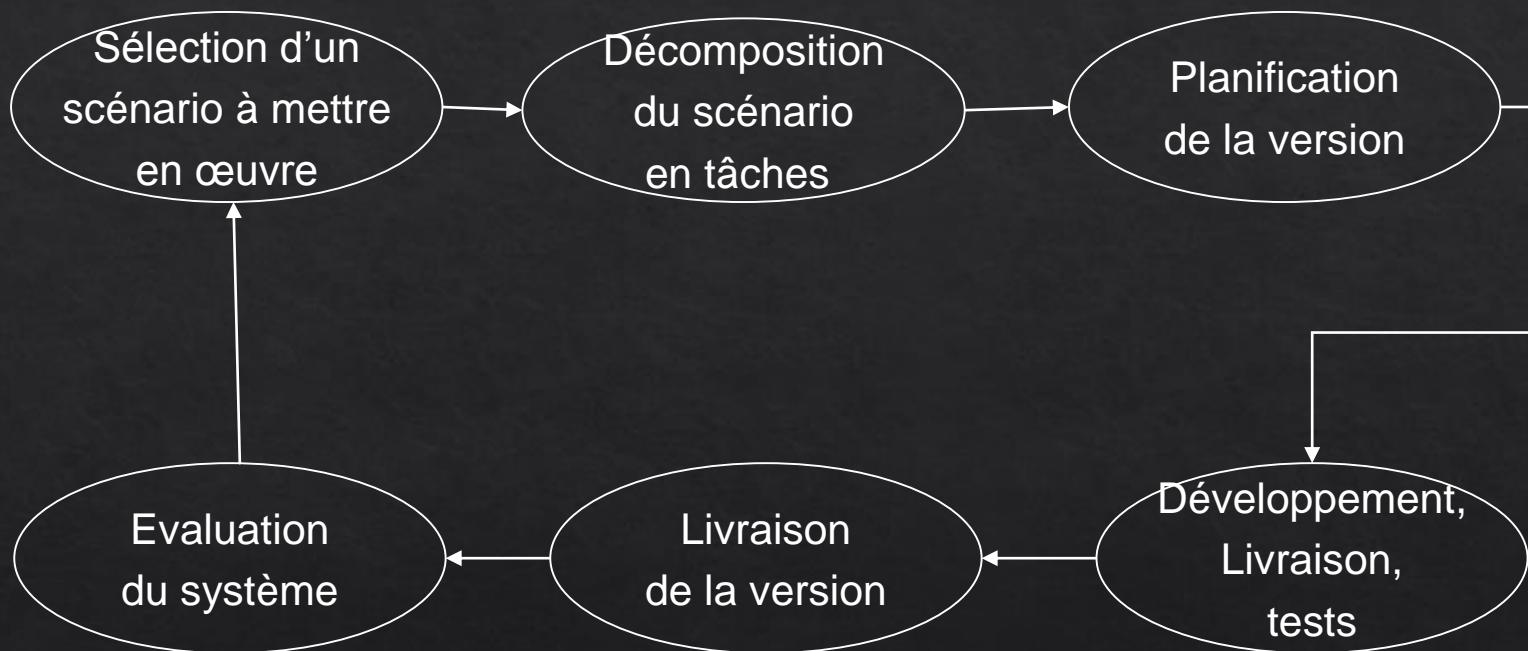
# Le manifeste agile -Principes

- ❖ Satisfaction des clients
- ❖ Accepter le changement du besoin
- ❖ Livraison fréquentes
- ❖ Implication du client
- ❖ Motivation des équipes
- ❖ Le dialogue face à face
- ❖ Opérationnel sinon rien
- ❖ Rythme soutenable
- ❖ Excellence technique
- ❖ La simplicité
- ❖ Equipes auto-organisées
- ❖ Amélioration continue

# Principes

- ❖ Ce sont des méthodes incrémentales
- ❖ Les activités de spécification, de conception et d'implantation sont entrelacées; on travaille avec des incréments de petite taille
- ❖ Le système est développé comme une succession de versions qui correspondent à l'ajout des incréments
- ❖ On prototype rapidement les interfaces du système pour avoir un retour rapide des utilisateurs

# « Extreme programming »



# XP – Terminology (a)

Principle or practice	Description
Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

# XP – Terminology (b)

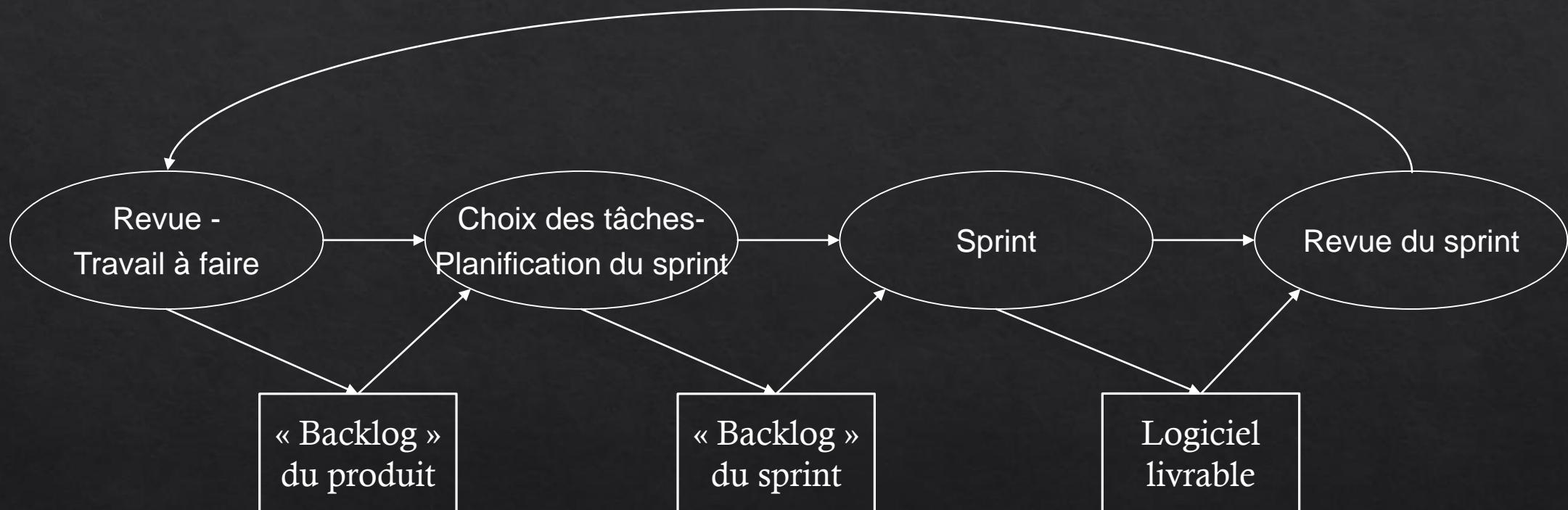
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the <i>Customer</i> ) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

# Scrum

*(Kent Schwaber et Jeff Shutterland)*

- ❖ Scrum est un cadre de travail agile qui a pour but de résoudre des problèmes complexes
- ❖ Scrum est une méthode agile qui se concentre sur le gestion itérative de projet en exploitant les propriétés créatives des membres de l'équipe de développement.
- ❖ Scrum se définit comme un cadre de développement permettant d'organiser la production d'un logiciel en privilégiant la collaboration entre les différents acteurs et en visant plutôt la satisfaction du client que le respect strict d'un cadre contractuel matérialisé par une documentation.
- ❖ On peut distinguer trois phases :
  - ❖ La phase initiale au cours de laquelle les fonctionnalités du système sont listés et une architecture logicielle générale est définie
  - ❖ Suit une série de “sprints”, chaque sprint correspondant à un incrément du système
  - ❖ La phase de terminaison du projet développe les derniers artefacts (manuel d'utilisation ...) et tire les leçons apprises durant le développement.

# Cycle de vie d'un sprint SCRUM



# Scrum terminology (a)

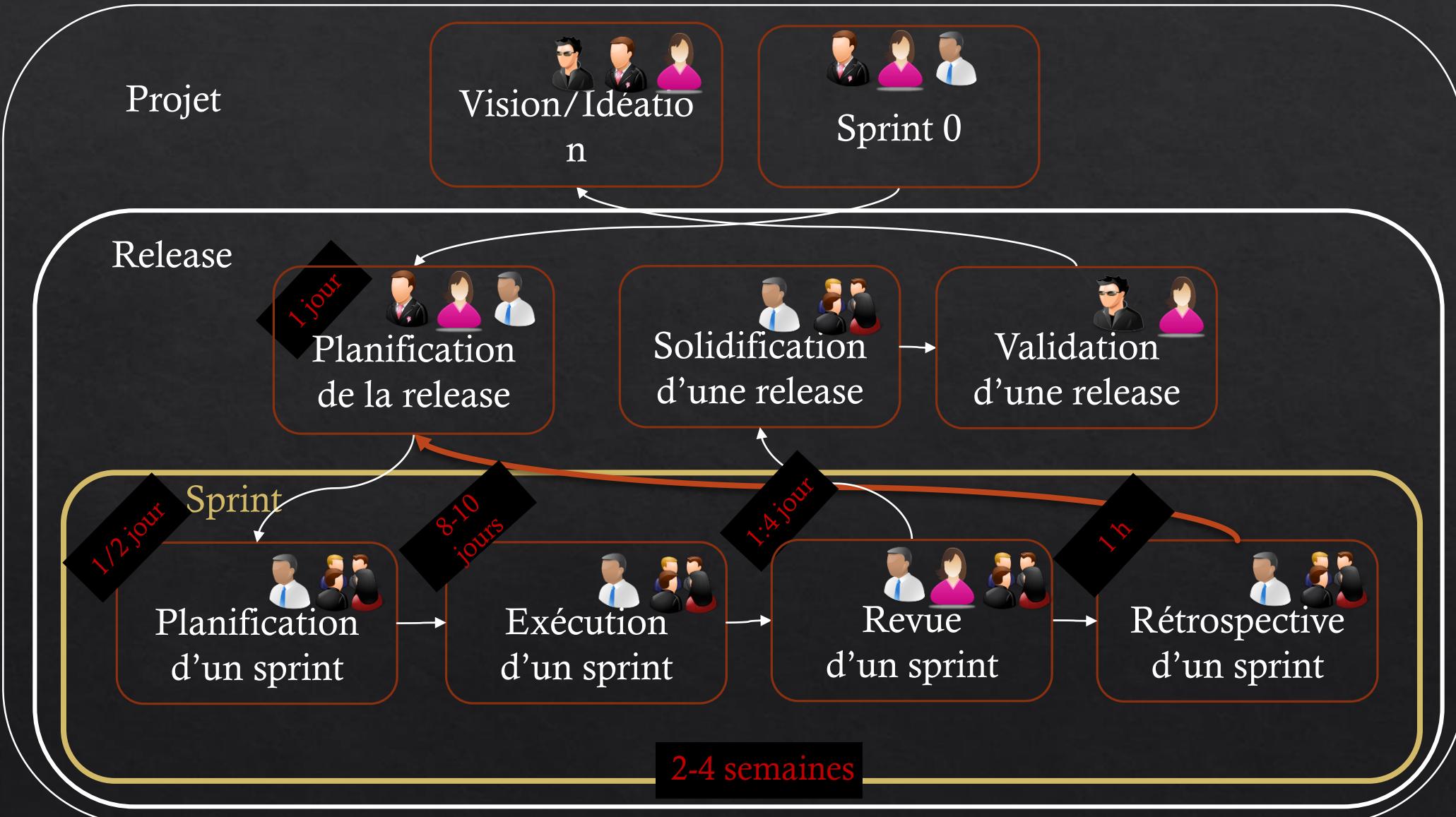
Scrum term	Definition
Development team (Equipe de développement)	Un groupe auto-organisateur de développeurs de logiciels, qui ne devrait pas être plus de 7 personnes. Ils sont responsables du développement du logiciel et d'autres documents de projet essentiels.
Incrément de produit potentiellement expédiable	L'incrément logiciel qui est livré à partir d'un sprint. L'idée est que cela devrait être « potentiellement expédiable », ce qui signifie qu'il est dans un état fini et aucun autre travail, comme les essais, est nécessaire pour l'intégrer dans le produit final. Dans la pratique, ce n'est pas toujours réalisable.
Product backlog (Backlog de produits)	Il s'agit d'une liste d'éléments à faire que l'équipe Scrum doit aborder. Il peut s'agir de définitions de fonctionnalités pour le logiciel, d'exigences logicielles, d'histoires d'utilisateurs ou de descriptions de tâches supplémentaires nécessaires, telles que la définition d'architecture ou la documentation utilisateur.
Product owner (Propriétaire de produit)	An individual (or possibly a small group) whose job is to identify product features or requirements, prioritize these for development and continuously review the product backlog to ensure that the project continues to meet critical business needs. The Product Owner can be a customer but might also be a product manager in a software company or other stakeholder representative.

# Scrum terminology (b)

Scrum term	Definition
Scrum	Une réunion quotidienne de l'équipe Scrum qui passe en revue les progrès réalisés et priorise le travail à faire ce jour-là. Idéalement, il devrait s'agir d'un court face-à-face qui inclut toute l'équipe.
ScrumMaster	ScrumMaster est responsable de s'assurer que le processus Scrum est suivi et guide l'équipe dans l'utilisation efficace de Scrum. Il est responsable de l'interfaçage avec le reste de l'entreprise et de s'assurer que l'équipe Scrum n'est pas détournée par des interférences extérieures. Les développeurs scrum sont catégoriques sur le fait que le ScrumMaster ne doit pas être considéré comme un chef de projet. D'autres, cependant, peuvent ne pas toujours trouver facile de voir la différence.
Sprint	Une itération de développement. Les sprints durent généralement de 2 à 4 semaines.
Velocity	Une estimation de l'effort de retard de produit qu'une équipe peut couvrir en un seul sprint. Comprendre la vitesse d'une équipe les aide à estimer ce qui peut être couvert dans un sprint et fournit une base pour mesurer l'amélioration des performances.

# Le processus logiciel SCRUM

# Processus logiciel Scrum



Client- User



Manager



Product owner

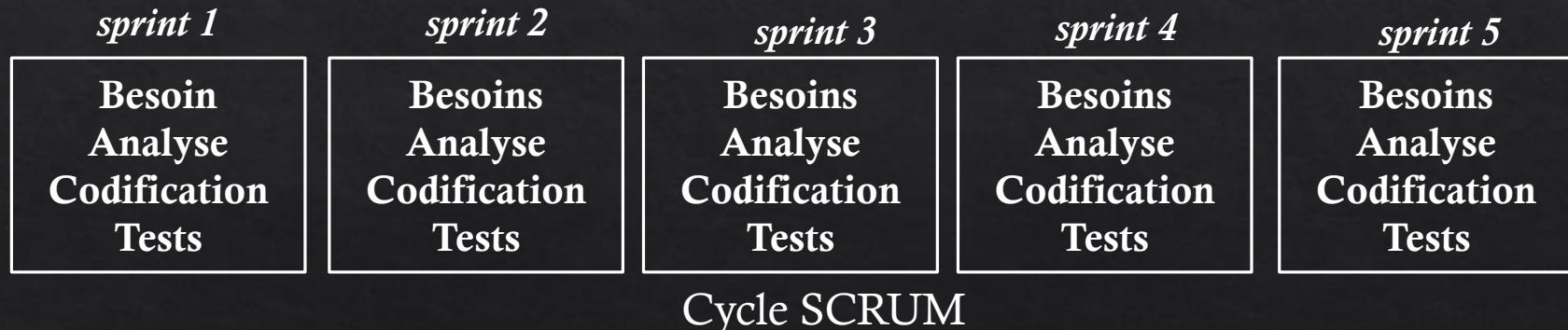
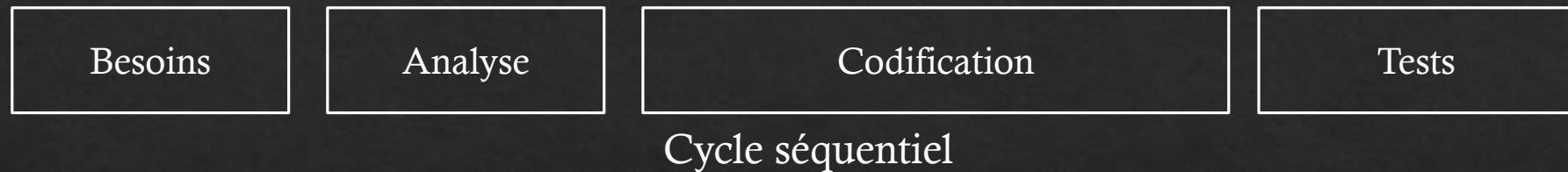


Scrum master



Développeur

# Processus logiciel SCRUM (*sprints* et *releases*)



# Processus logiciel SCRUM

- ❖ Une phase d'idéation
- ❖ Une succession de sprints de taille fixe
- ❖ Chaque *sprint* produit un logiciel fonctionnel
- ❖ Chaque *sprint* réalise toutes les activités de développement logiciel
- ❖ Les *sprints* sont groupés en *Releases* (de taille fixe)

# *Release SCRUM vs. Release traditionnelle*

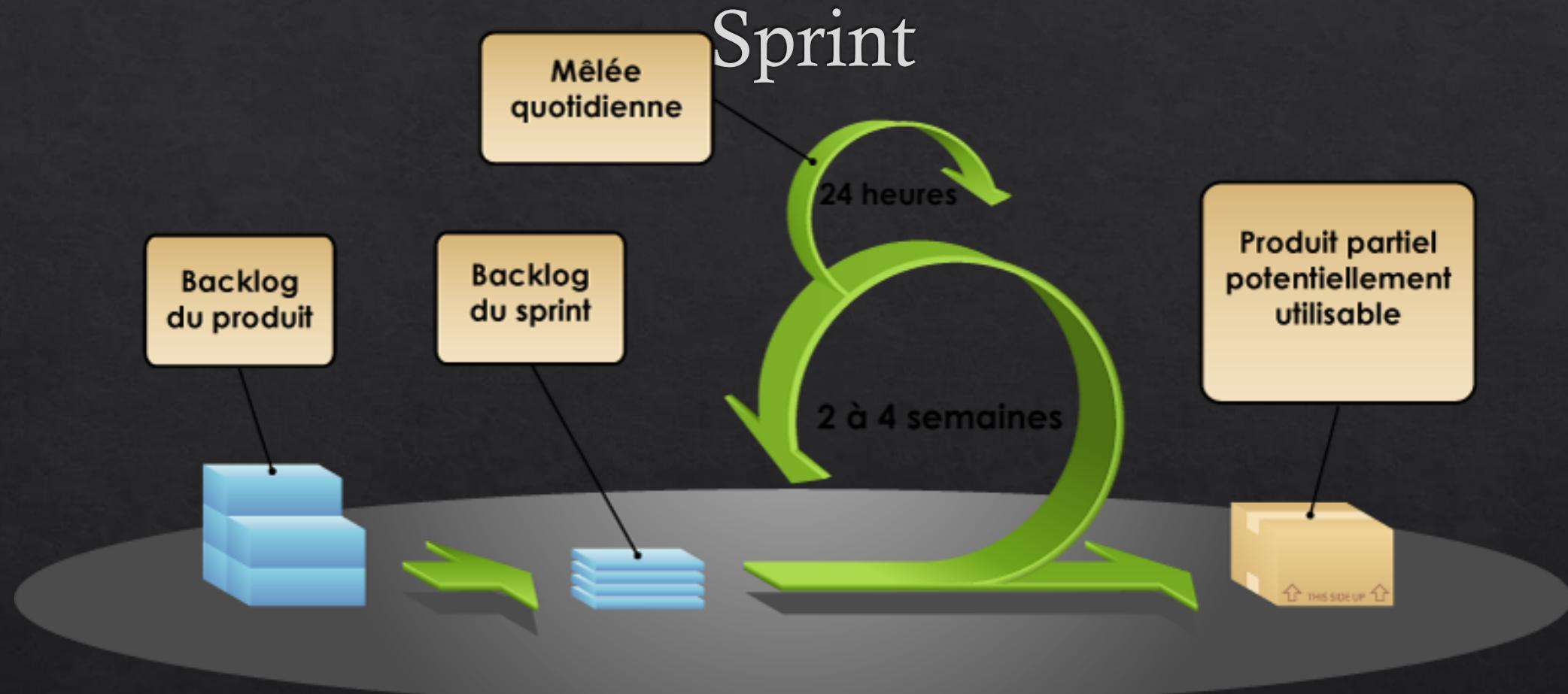
- ❖ Traditionnellement, une *release* (livrable) correspond à une nouvelle version d'un logiciel
  - ❖ Incrément fonctionnel (une *release* correspond à la réalisation d'un objectif fonctionnel)
  - ❖ Evolution technique
- ❖ SCRUM
  - ❖ Une *release* :
    - ❖ correspond à la livraison d'une *feature* (fonctionnalité)
    - ❖ travail réalisé pendant une période de temps fixe qui comprend plusieurs *sprints*
    - ❖ pour faire coïncider ces deux objectifs, on décompose/regroupe les *features*
  - ❖ Mais une nouvelle version du produit (livrable) peut-être produite à la fin de n'importe quel sprint

# Périodes d'une *Release*

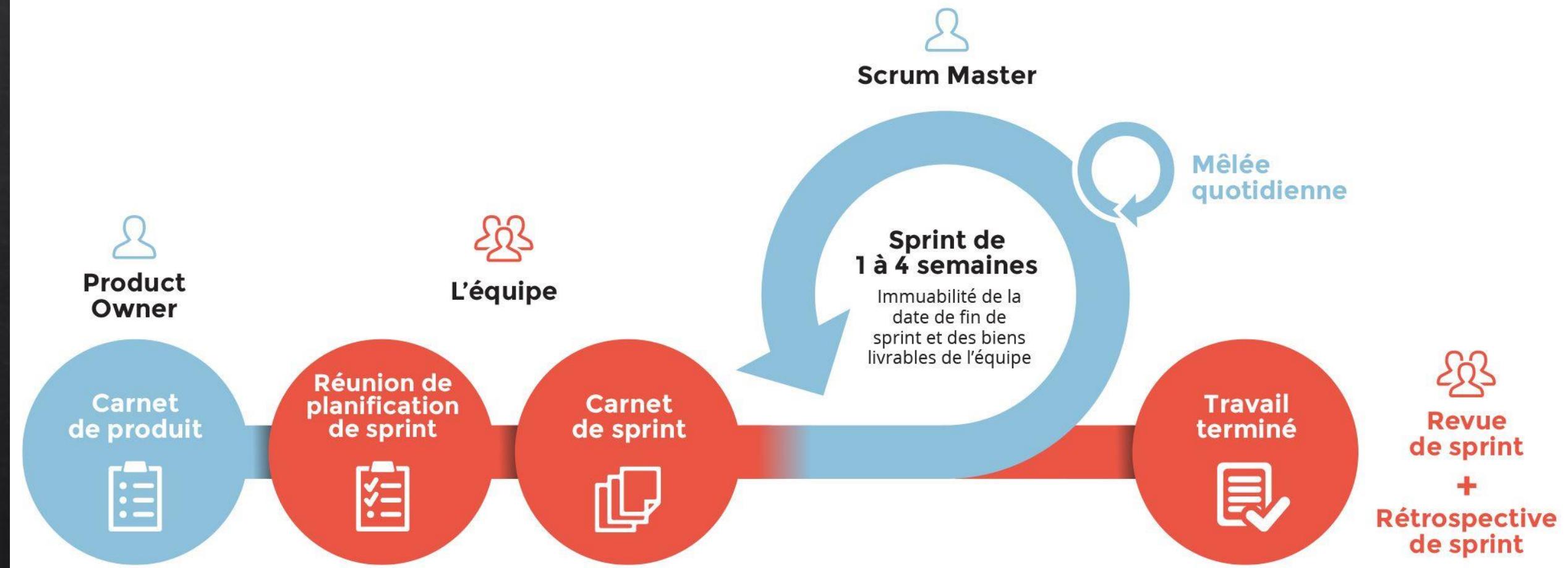
- ❖ La période avant le premier *sprint* de développement, appelée « sprint zéro » : planification de la *release* en plusieurs *sprints*
- ❖ La période des *sprints* (de développement)
- ❖ La période après le dernier *sprint* et avant la fin de la release

# *Sprint*

- ❖ Chaque *release* est décomposée en *sprints* au cours de l'activité de planification de la *release* (la première fois « *sprint zéro* »)
- ❖ Un *sprint* est une période de développement de taille fixe (en moyenne 2 à 4 semaines)
  - ❖ Durée fixe, équipe stable
  - ❖ Un *sprint* se décompose en un ensemble de tâches
  - ❖ Mêlée journalière : réunion d'équipe pour faire le point sur le travail réalisé depuis le début du sprint et le travail à réaliser avant la fin du sprint
  - ❖ Chaque *sprint* termine par :
    - ❖ une revue du produit
    - ❖ une rétrospective sur le processus



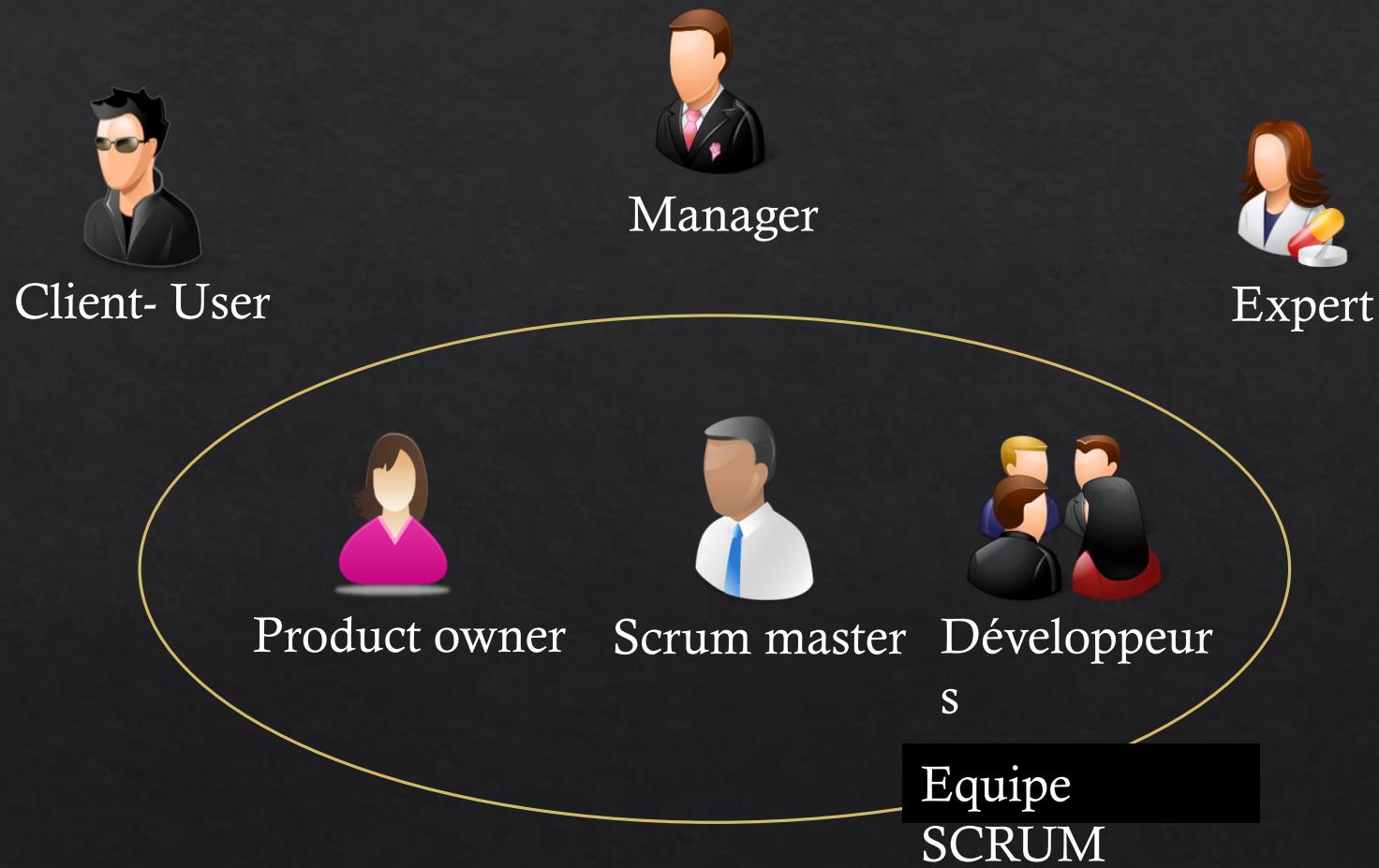
*Backlog*: carnet, liste ordonnée de « choses » à faire  
(*user stories*, tâches)



# *Les acteurs*

Des créateurs de valeur

# Rôles



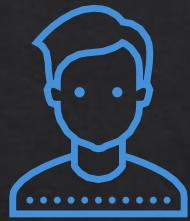
# L'équipe

- ❖ Composition:
  - ❖ 1 Product Owner
  - ❖ 1 Scrum Master
  - ❖ 2 à 7 développeurs
    - ❖ Le product owner et le Scrum master peuvent aussi prendre le rôle de développeur
- ❖ Principes
  - ❖ Auto-organisation
  - ❖ Pluridisciplinarité
  - ❖ Stabilité
  - ❖ Valeurs communes

# *Product owner*

- ❖ Responsabilités

- ❖ Fait partager la vision globale du produit
- ❖ Gère le backlog du produit (liste ordonnée des « choses » à faire)



Utilisateur



Action



But

- ❖ Définit les priorités
- ❖ Accepte ou rejette les *Releases* (livrables)

# *Product owner*

- ❖ Caractéristiques
  - ❖ Communication
  - ❖ Vision
  - ❖ Leadership
  - ❖ Disponibilité



# Scrum master

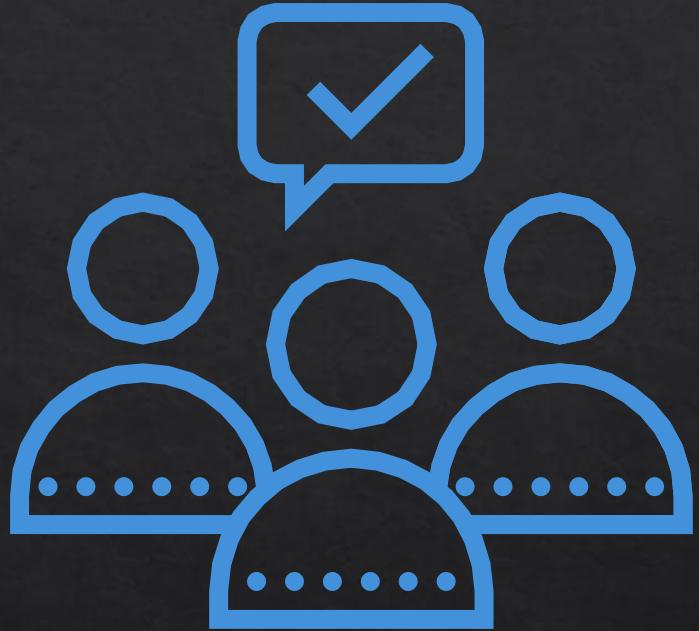
## ❖ Responsabilités

- ❖ n'est pas « le chef », mais un facilitateur
- ❖ Garantir le processus
- ❖ Motiver l'équipe
- ❖ Faire appliquer les bonnes pratiques de Scrum
- ❖ Gérer les obstacles
- ❖ Est l'arbitre qui aide le groupe à respecter les règles
- ❖ Animer les rituels
- ❖ Lever les freins



# Scrum master

- ❖ Caractéristiques
  - ❖ Animation
  - ❖ Empathie
  - ❖ Rigueur
  - ❖ Discernement
  - ❖ Autorité



# Scrum master *vs* Chef de projet

- ❖ Chef de projet
  - ❖ Responsable du projet
  - ❖ Ascendant hiérarchique
  - ❖ Prise de décision
- ❖ Scrum Master
  - ❖ Responsabilité partagée
  - ❖ Interlocuteur
  - ❖ Accompagner vers la décision



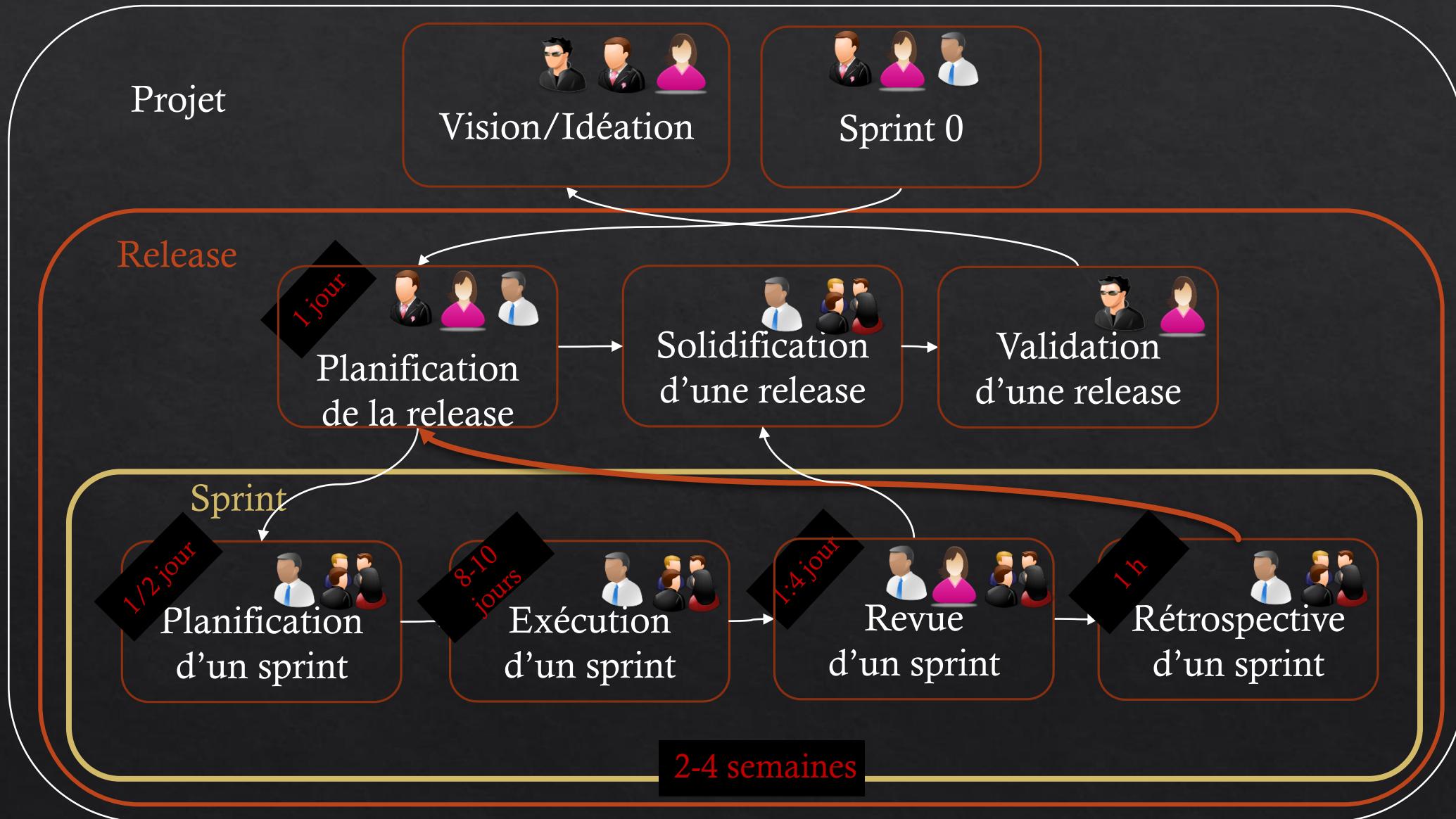
# *Développeur*

- ❖ Veille à la qualité technique
- ❖ Le développeur s'auto-organise
- ❖ Scrum ne précise rien sur les techniques de développement à mettre en oeuvre



0101  
1001

# Processus logiciel Scrum



Client- User



Manager



Product owner



Scrum master



Développeur

# Les objets de SCRUM

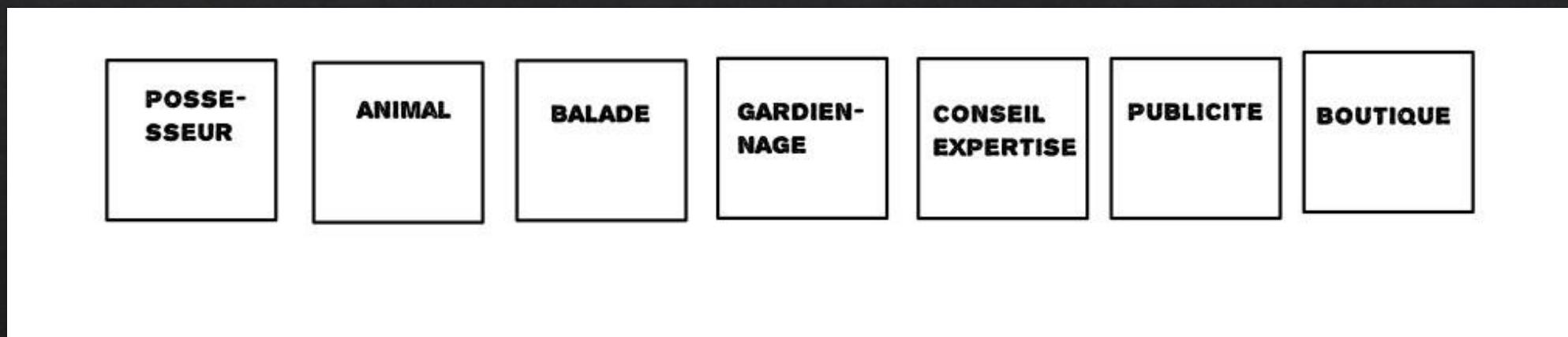
## Feature, Story, Tâche, Backlog

# *Feature*

- ❖ Une *feature (fonctionalité)* est un service ou une fonctionnalité du produit à développer
- ❖ Elle se décompose en *stories (histoires)* de tailles différentes.  
On distingue :
  - ❖ les *stories complexes (épiques)* qui seront affinées en *stories plus simples*
  - ❖ Les *stories atomiques* qui ne se décomposent pas et sont réalisées dans les *sprints*

# *Feature* (exemple)

un site pour propriétaires d'animaux domestiques  
Des *features* (fonctionnalités, chapites ...)



# Workflow d'une *feature*

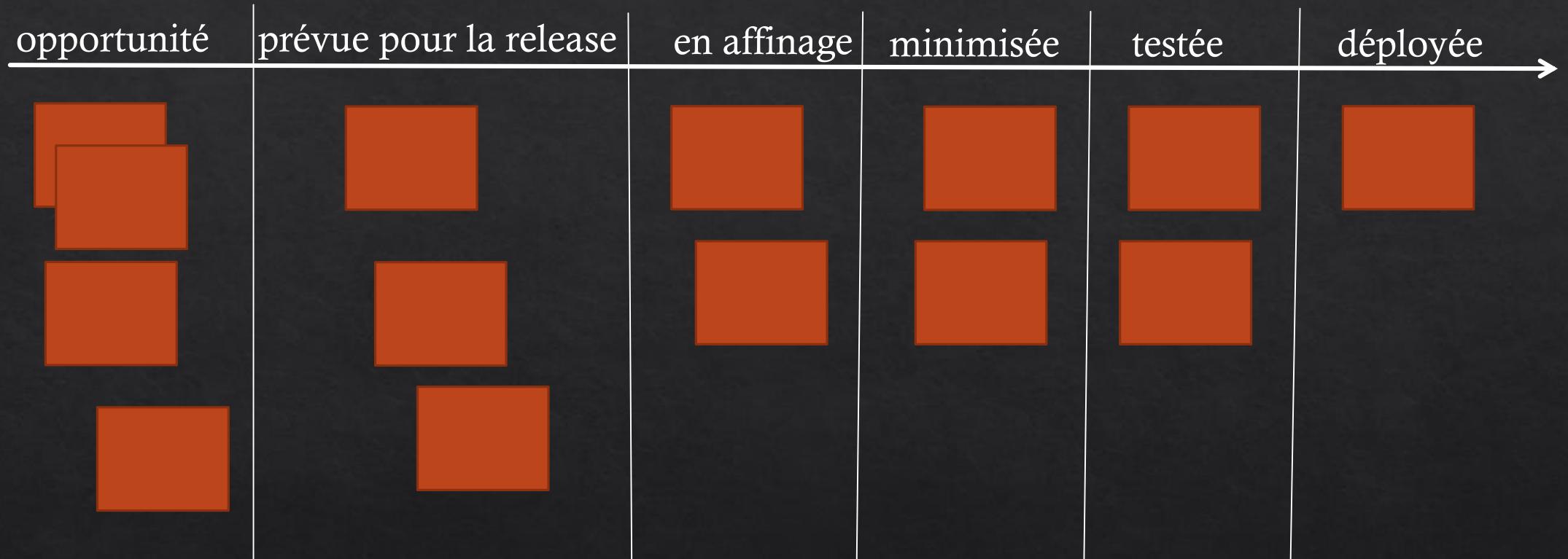
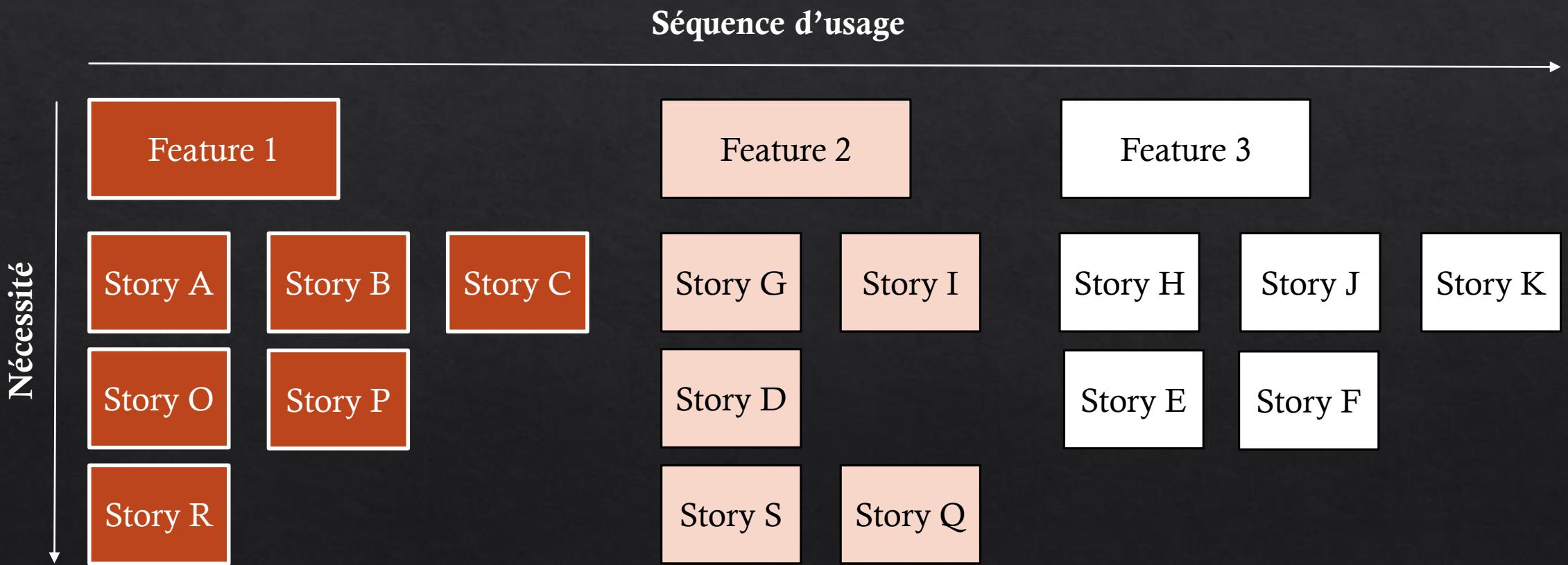


Tableau des *features*

# Workflow d'une *feature*

- ❖ Opportunité : idée de feature qui présente une opportunité pour le produit
- ❖ Prévue pour la release : l'étude d'opportunité a abouti, et la feature est prévue pour la release en cours
- ❖ En affinage : initialement sous forme d'épique, est décomposée en stories
- ❖ Minimisée : *Minimal Marketable Feature*
- ❖ Testée : ... avec des stories finies
- ❖ Déployée : utilisable par des utilisateurs qui peuvent fournir du feedback

# *Story map :* décomposition d'une *feature* en *stories*



<b>POSSE-SSEUR</b>	<b>ANIMAL</b>	<b>BALADE</b>	<b>GARDIEN-NAGE</b>	<b>CONSEIL EXPERTISE</b>	<b>PUBLICITE</b>	<b>BOUTIQUE</b>
Données Basiques	Données Basiques	Liste simple	Liste simple	fiches conseils vétérinaires	Encart pub	Catalogue
Données Avancées	Pédigree	Liste départs sur carte geoloc	Créer/mod gardiennage	fiches conseils éleveurs	Compteurs / stats	Panier
Ajout Média	Ajout média	Liste participants	Note organisateur	Forum	Mailing	Paiement CB
Messagerie Tchat	Données avancées	Offrir/créer balade	Commentaires	Tchat expert	Profiling	Paiement Paypal
Archivage		Visu parcours sur carte	Ajout média		Invitation événement	Paiement 3 fois
géolocalisation		Note organisateur				Coupon promo
		Commentaires				Promo avancée
		Notifications				Visu export/facture



# *Story*

- ❖ Une *story* est une exigence du système à développer, formulée en une ou deux phrases dans le langage de l'utilisateur.
- ❖ Les *Stories* émergent au cours d'ateliers de travail menés avec le Métier, le Client et/ou les Utilisateurs.
- ❖ On distingue :
  - ❖ *Story* fonctionnelle
  - ❖ *Story* technique
  - ❖ Correction de bug
  - ❖ Remboursement de la « dette technique »

# *Story*

- ❖ Les 3C

- ❖ Carte : l'histoire est courte et sa description tient sur une carte (demi-page)
- ❖ Conversation : l'histoire est définie avec les gens du métier
- ❖ Confirmation : l'histoire est confirmée par des tests d'acceptation rédigés au même moment que celle-ci

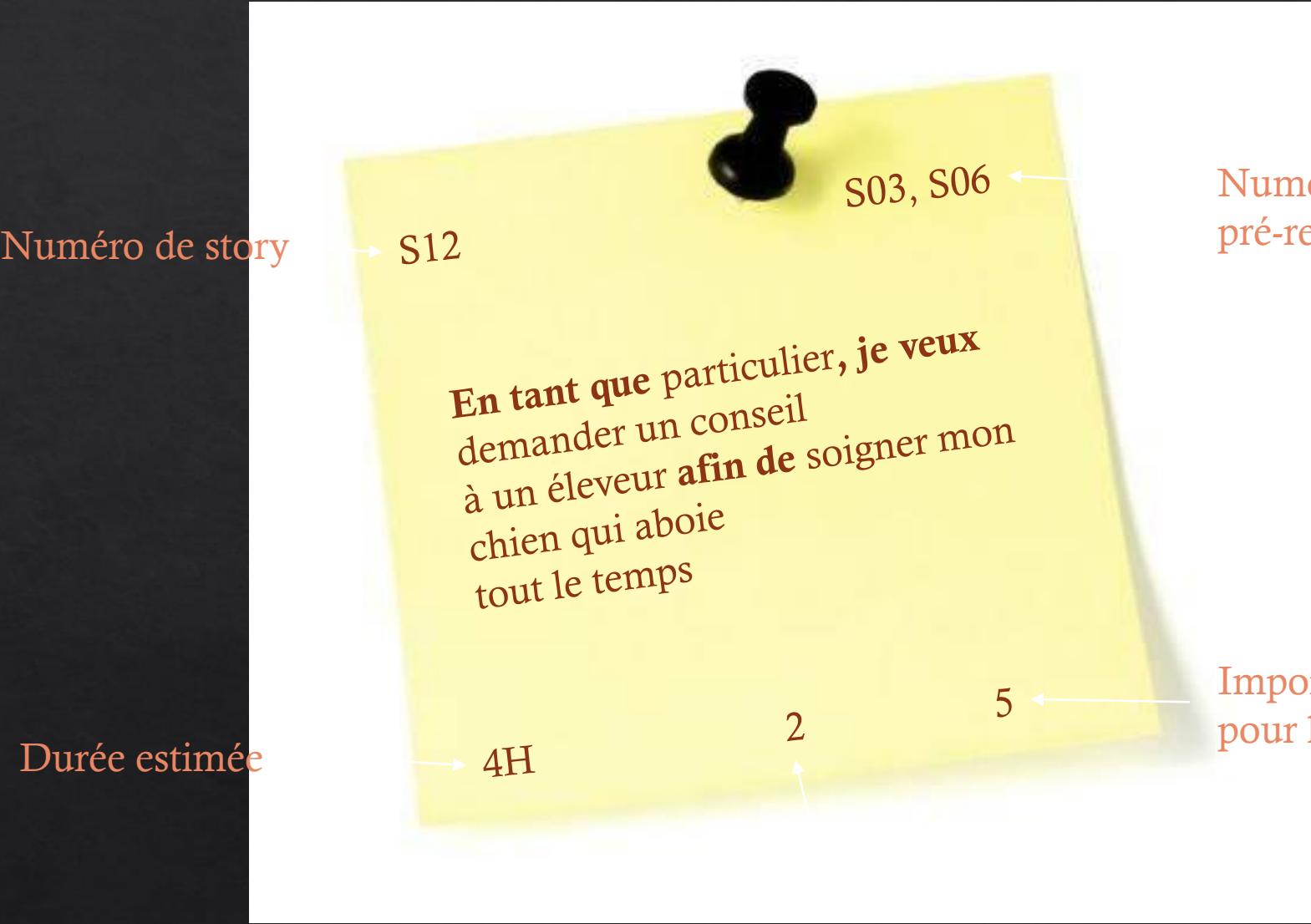
- ❖ Workflow de la *story*

- ❖ Idée d'une *story* rédigée sur une **Carte** (1/2 feuille)
- ❖ **Conversation** dirigée par le *product owner* qui inclut les gens du métier
- ❖ L'équipe apporte sa **Confirmation** que la *story* est prête
- ❖ L'équipe réalise la *story*
- ❖ Le *product owner* apporte sa **Confirmation** que la *story* est finie

# *Description type d'une story*

- ❖ Plan type
  - ❖ En tant que <acteur>, je veux <un but> [afin de <une justification>]
    - ❖ En tant que client, je veux pourvoir accéder au site de ma banque afin de gérer mon compte sur Internet
- ❖ Priorité
- ❖ Nombre de points
  - ❖ Représente le niveau de difficulté intrinsèque, en fonction de la taille et de la valeur métier
    - ❖ Corrélées en moyenne, mais pas toujours (en fonction d'une forte valeur métier, de la réutilisation possible de composants ...)
- ❖ Conditions d'acceptation
  - ❖ Etant donné <le contexte> quand je <événement> alors <résultat>
    - ❖ Etant donné que je suis sur la page de connexion et que j'ai entré un login et un mot de passe dans le formulaire et que le login et le mot de passe correspondent à un utilisateur enregistré, quand je clique sur le bouton “Se connecter” alors j'arrive sur la page d'accueil du site.

# *Post-it de story*

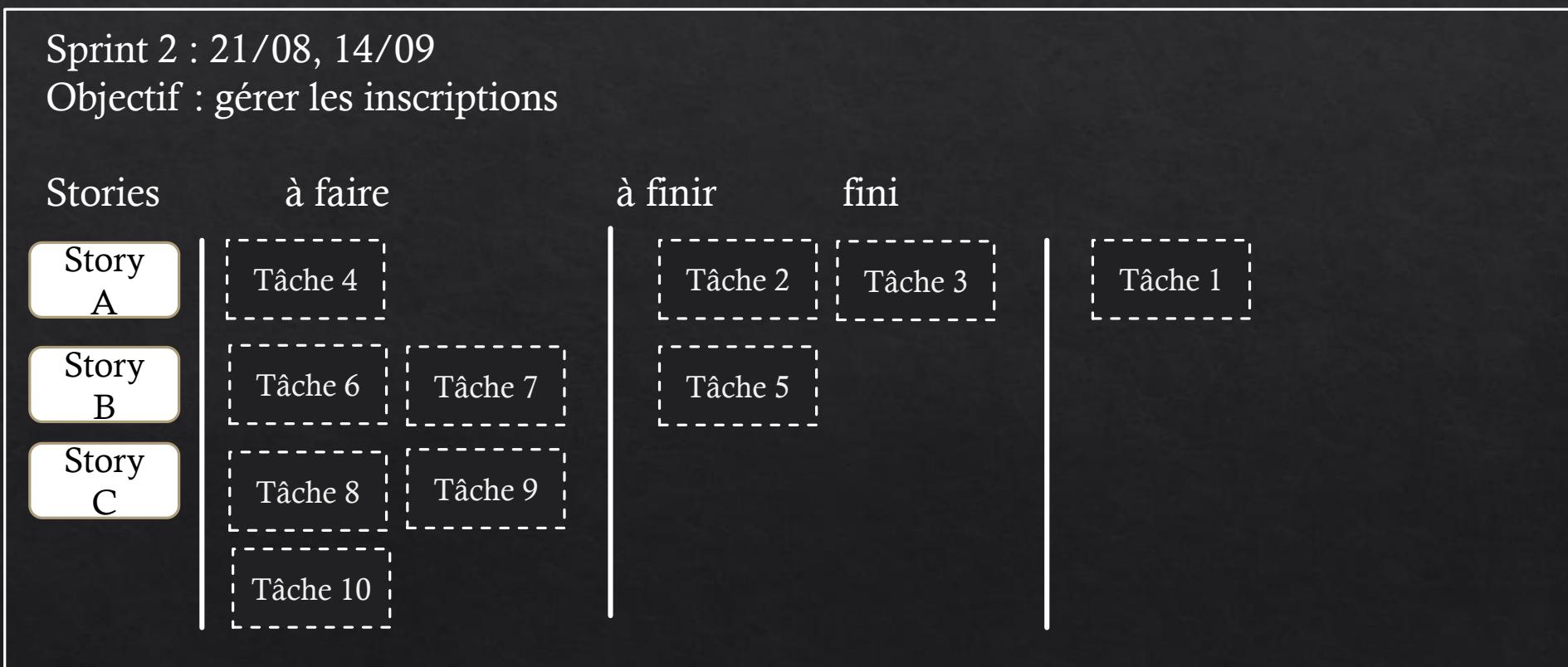


# Estimation des points d'une story

- ❖ Difficulté intrinsèque
  - ❖ Taille et valeur métier
    - ❖ Corrélées en moyenne, mais pas toujours (en fonction d'une forte valeur métier, de la ré-utilisation de composants ...)
- ❖ Planning Poker (source Wikipedia)
  - ❖ Les participants s'installent autour d'une table, placés de façon que tout le monde puisse se voir.
  - ❖ Le responsable de produit explique à l'équipe un scénario utilisateur (*user story*).
  - ❖ Les participants posent des questions au responsable de produit, discutent du périmètre du scénario, évoquent les conditions de satisfaction qui permettront de le considérer comme "terminé".
  - ❖ Chacun des participants évalue la complexité de ce scénario, choisit la carte qui correspond à son estimation et la dépose, face vers le bas, sur la table devant lui.
  - ❖ Au signal du facilitateur, les cartes sont retournées en même temps.
  - ❖ S'il n'y a pas unanimité, la discussion reprend.
  - ❖ On répète le processus d'estimation jusqu'à l'obtention de l'unanimité.
  - ❖ Une procédure optimisée consiste, après la première "donne", de demander aux deux acteurs ayant produit les évaluations extrêmes d'expliquer leurs points de vue respectifs. Ces explications achevées et comprises de tous, une nouvelle estimation est produite et c'est alors la moyenne arithmétique de ces estimations qui est prise en compte.

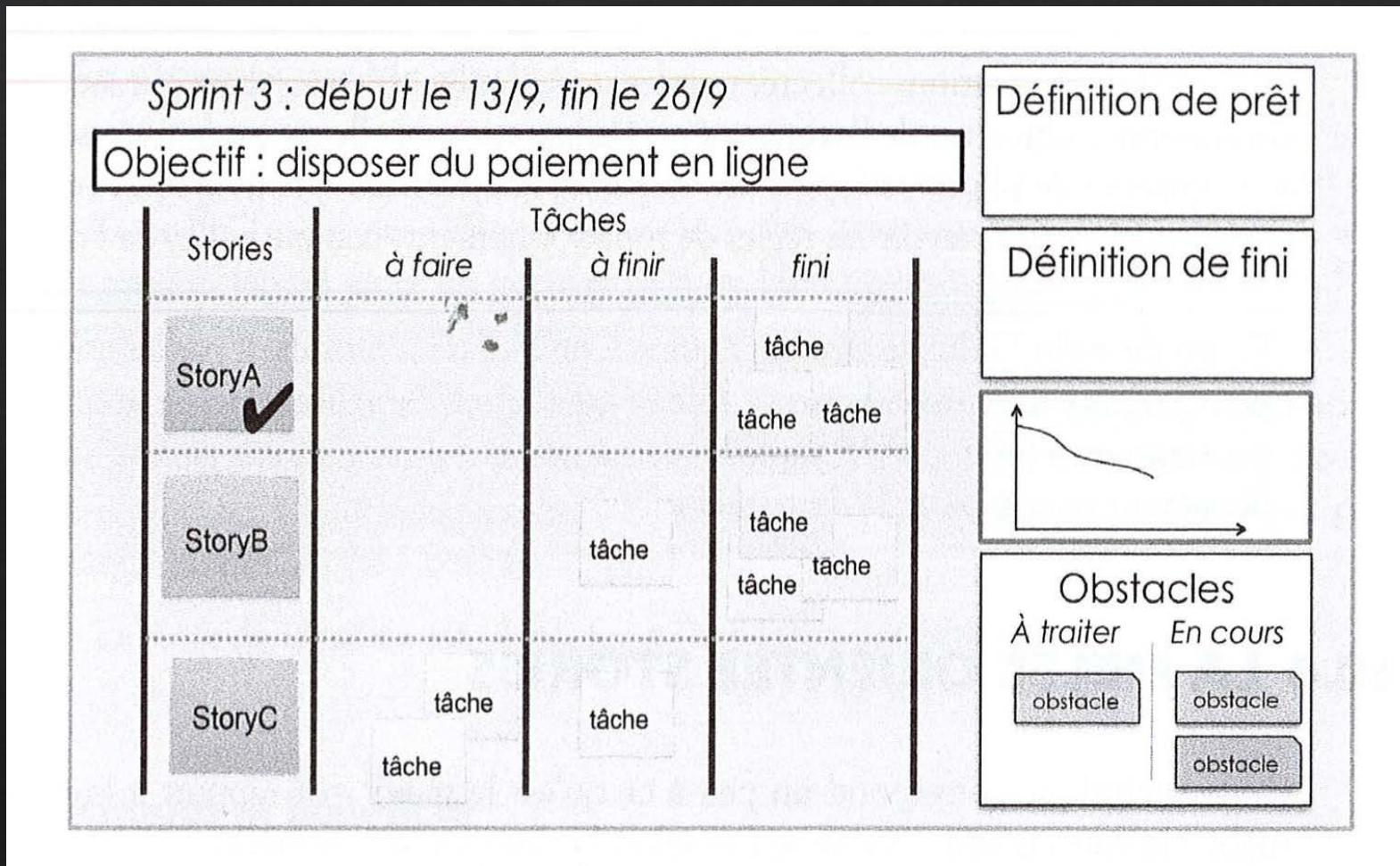
# Tâche

- A l'exécution, une story se décompose en tâches



# *Tableau de la Story*

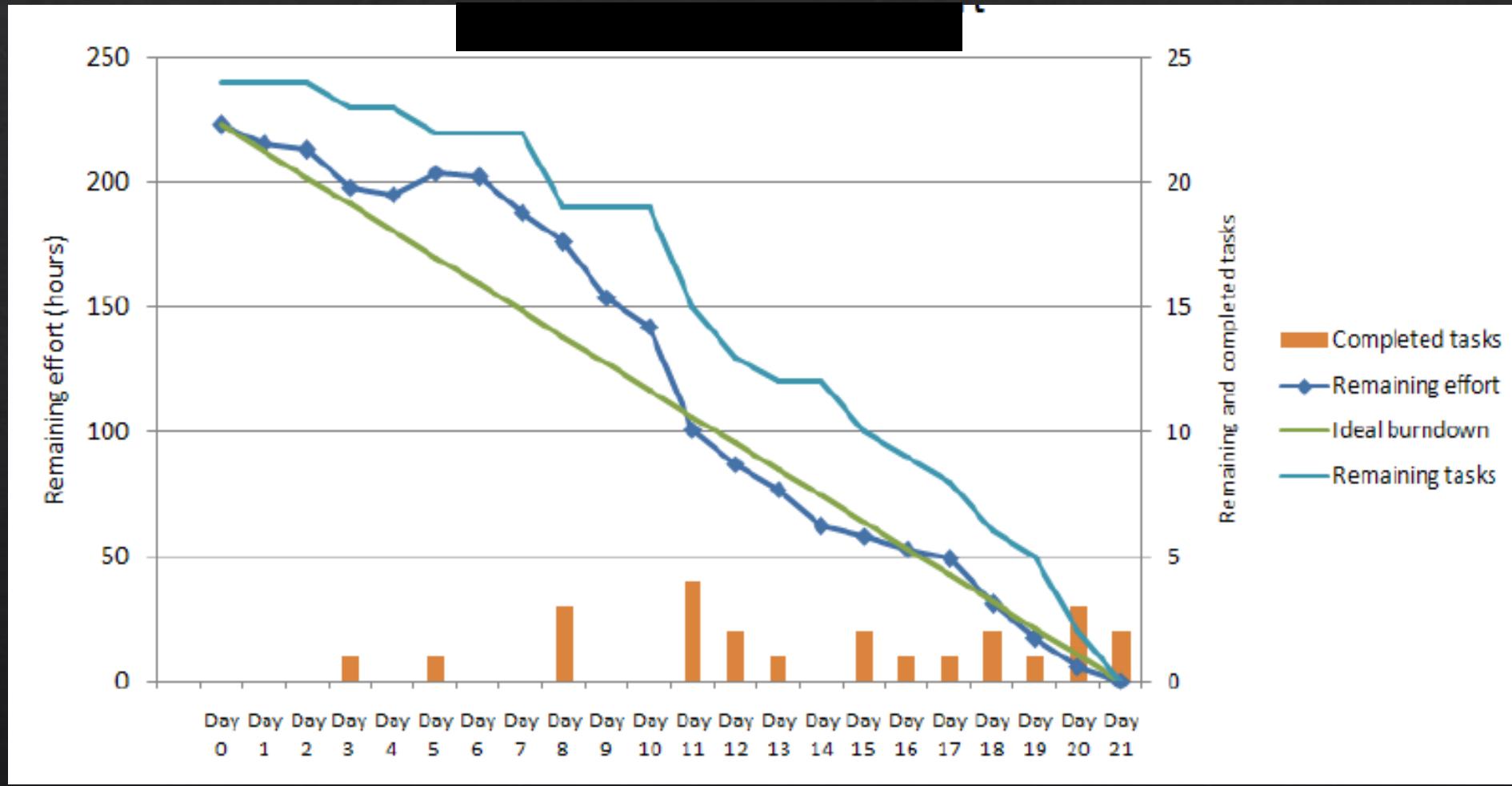
- ◆ Le tableau de la *story* décrit son état.



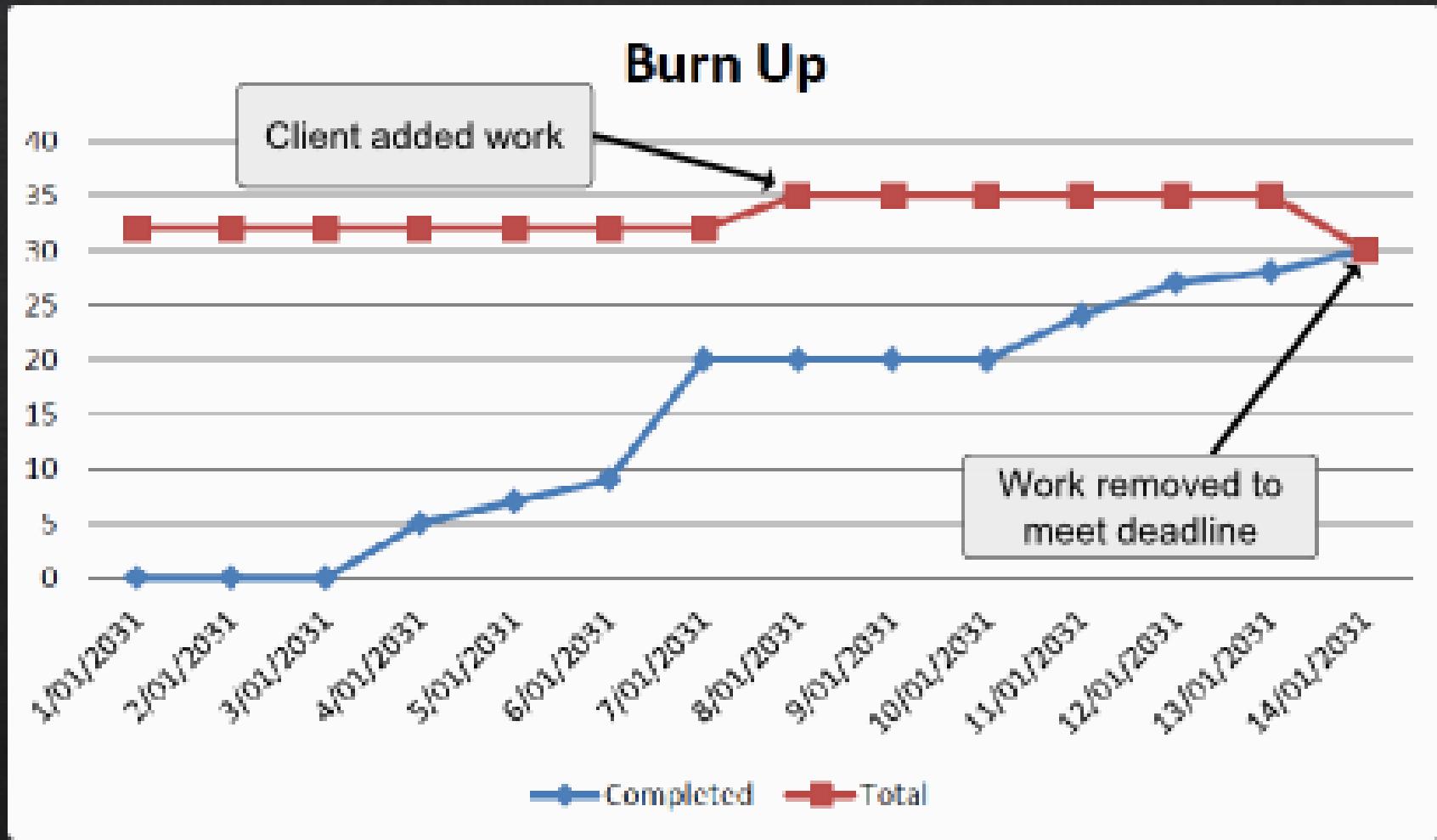
# Indicateurs

- ❖ Sprint
  - ❖ *Burndown de sprint* (orienté reste à faire)
  - ❖ *Burnup de sprint* (orienté ce qui a été déjà fait)
- ❖ Release
  - ❖ *Burndown de release*
  - ❖ *Burnup de release*
- ❖ Equipe
  - ❖ Vélocité (capacité de l'équipe)
  - ❖ Suivi des obstacles
    - ❖ Perturbations exogènes ou endogènes qui perturbent le bon déroulement du sprint
    - ❖ Peut de générer de nouvelles tâches (dette) et/ou leur réorganisation

# *Burndown graph*



# Burnup graph



# Vélocité

- ❖ Estime la capacité de l'équipe en nombre de points de stories par sprint
- ❖ Utilisé pour la planification de la release
- ❖ Affinée à la fin de chaque sprint
- ❖ Tendance à la stabilité

# Les activités propres à SCRUM

# Idéation



- ❖ Définition d'une vision commune
  - ❖ Identification des *features*
    - ❖ Impact mapping
  - ❖ Identification des parties prenantes
    - ❖ Acteurs
- ❖ Création d'un *backlog* de haut niveau du produit
  - ❖ Tableau ordonné des features
  - ❖ (éventuellement Story map haut niveau)

# Impact mapping

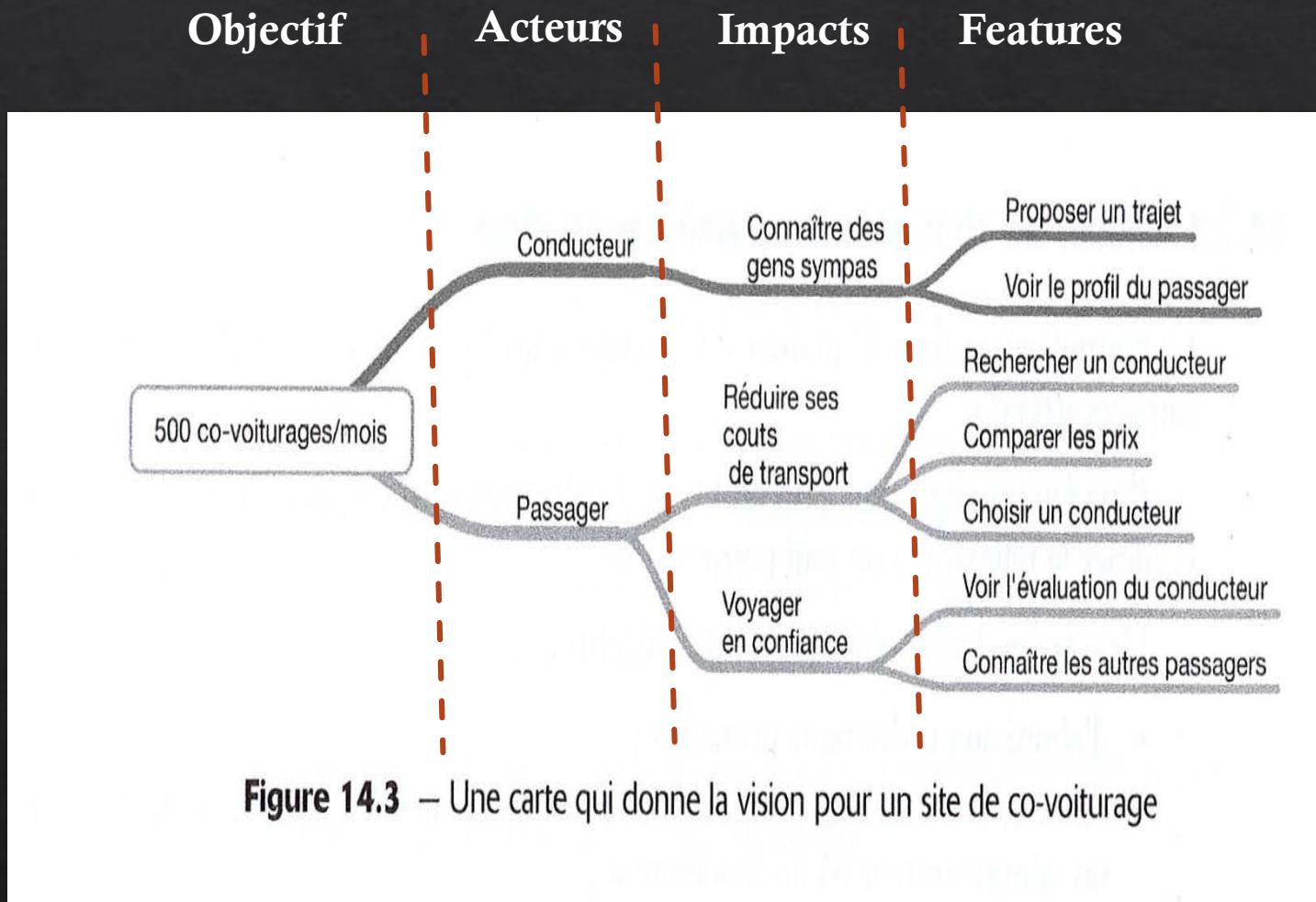


Figure 14.3 – Une carte qui donne la vision pour un site de co-voiturage

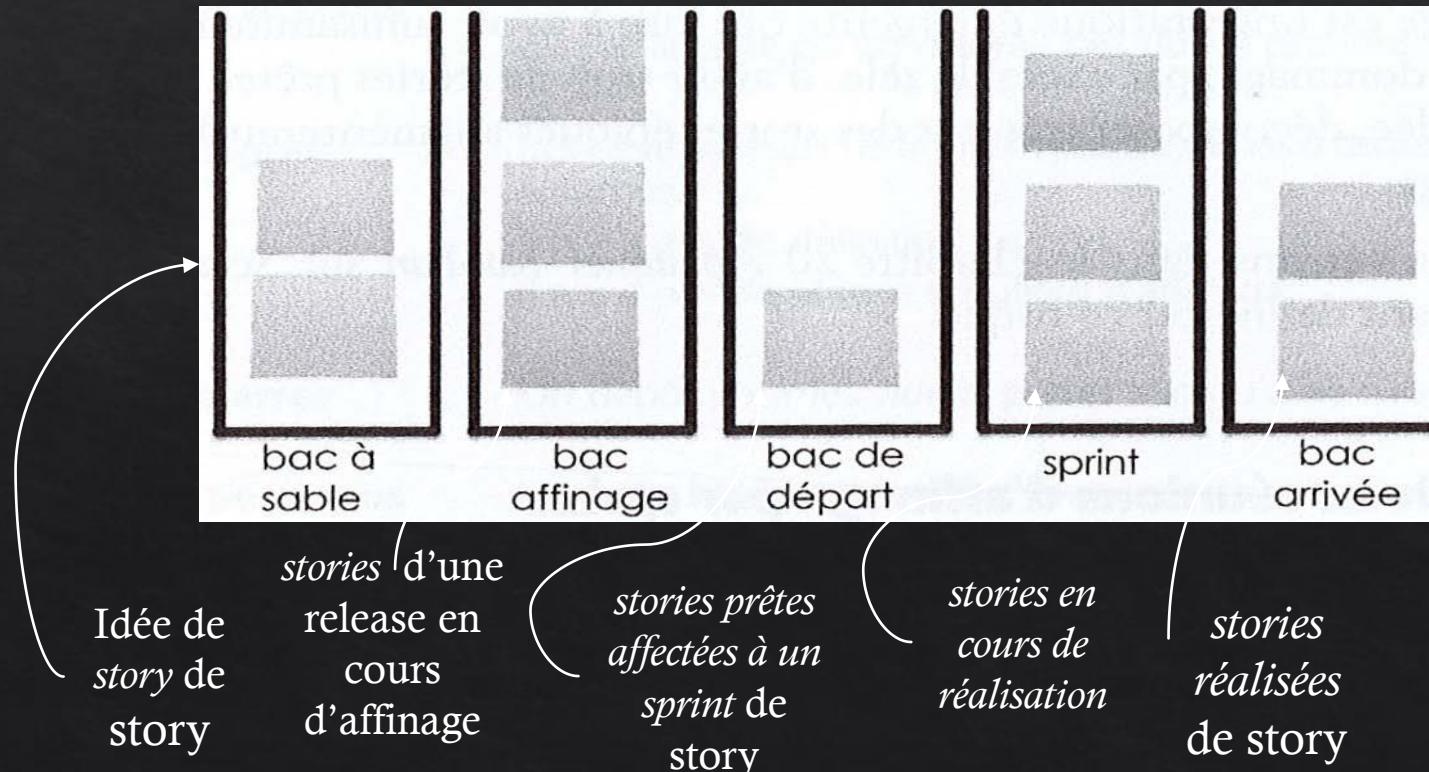
# *Sprint « Zéro »*



- ❖ Affinage du backlog
  - ❖ *Story mapping (feature → stories)*(partiel)
  - ❖ Description des stories (description, conditions d'acceptation)
  - ❖ Ordonnancement des stories
  - ❖ Planification de la première *release*
    - ❖ Approvisionnement du bac d'affinage
    - ❖ Approvisionnement du bac de départ du sprint 1
- ❖ Peut durer plusieurs journées

# Le *backlog* (carnet de route)

- ❖ Liste ordonnée des choses (stories) à faire
- ❖ En pratique, plusieurs (sous-)backlogs
  - ❖ *On peut distinguer le backlog de produit et le backlog de sprint*

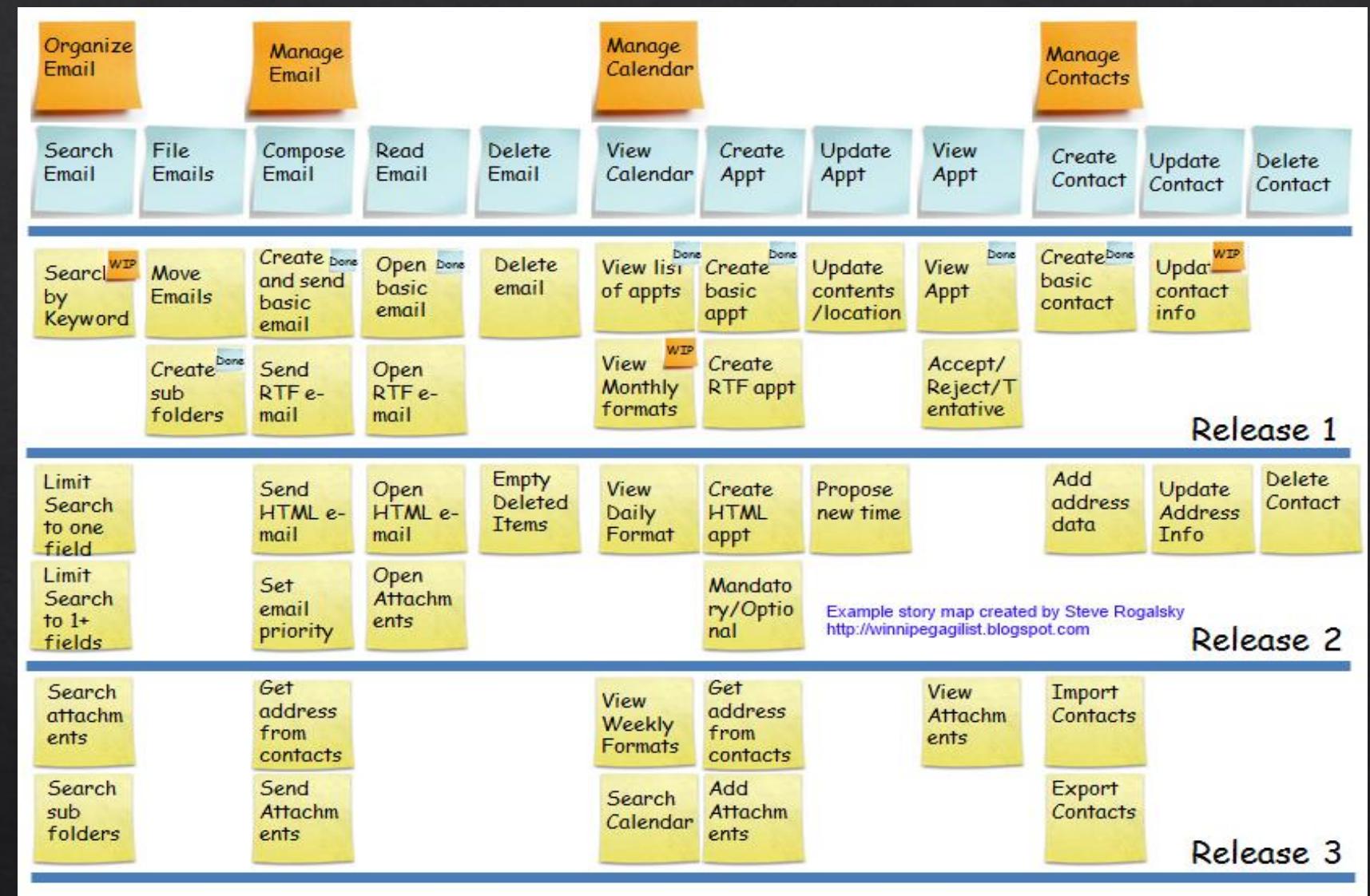


# Story mapping

- ❖ Ordonnancement des features
- ❖ Décomposition en stories
- ❖ (Organisation des releases)

POSSESSEUR	ANIMAL	BALADE	GARDIENNAGE	CONSEIL EXPERTISE	PUBLICITE	BOUTIQUE
Données Basiques	Données Basiques	Liste simple	Liste simple	fiches conseils vétérinaires	Encart pub	Catalogue
Données Avancées	Pédigree	Liste départs sur carte geoloc	Créer/mod gardiennage	fiches conseils éleveurs	Compteurs / stats	Panier
Ajout Média	Ajout média	Liste participants	Note organisateur	Forum	Mailing	Paiement CB
Messagerie Tchat	Données avancées	Offrir/créer balade	Comment -aires	Tchat expert	Profiling	Paiement Paypal
Archivage		Visu parcours sur carte	Ajout média		Invitation événement	Paiement 3 fois
géolocalisation		Note organisateur				Coupon promo
		Comment -aires				Promo avancée
		Notifications				Visu export/facture

<https://pablopernot.fr/2017/01/cartographie-plan-action/>

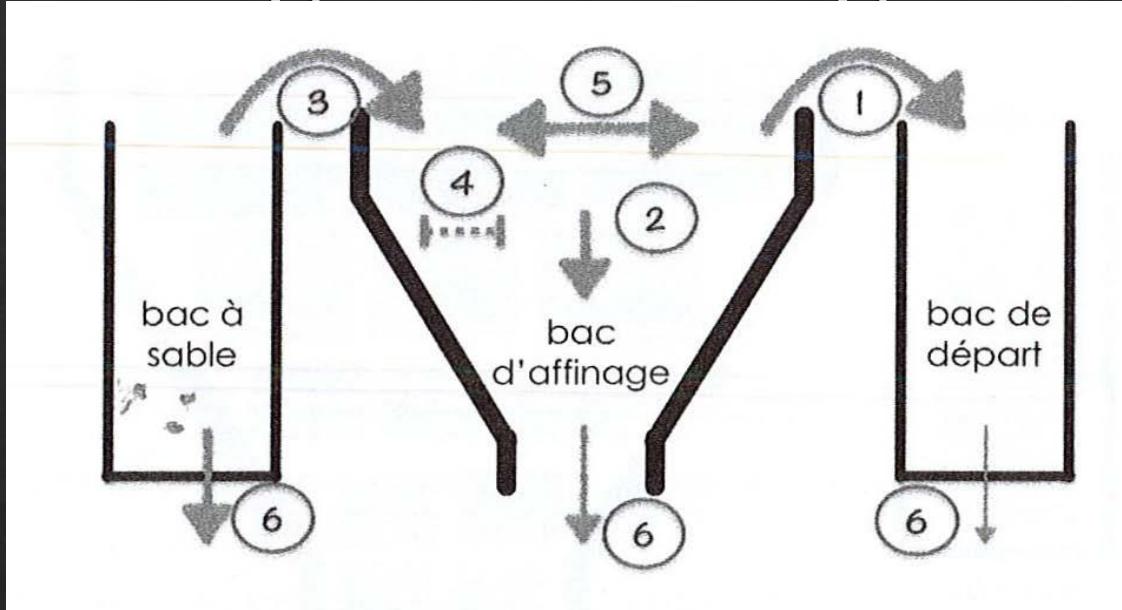


# Planification des *releases*



- ❖ Affiner les risques, les incertitudes en fonction des retours de la revue et de la rétrospective du dernier sprint
- ❖ Ajuster la vélocité de l'équipe (capacité de travail de l'équipe en nombre de points de story)
- ❖ Affiner, (re-)planifier le(s) prochain(s) sprint(s)
  - ❖ Définition de prêt et fini
  - ❖ Nombre de points
- ❖ Affiner, (re-)planifier la future release

# Affinage du backlog

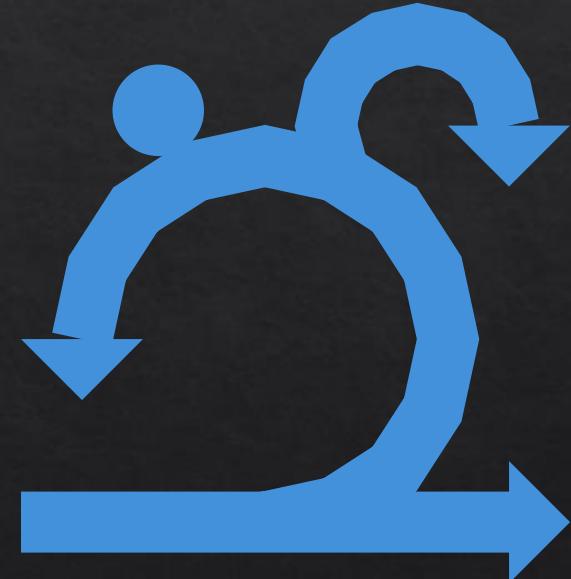


1. Approvisionner le bac de départ en stories prêtes
2. Identifier les *epics* à décomposer en stories simples
3. Identifier les stories du bac à sable qui peuvent entrer dans le bac d'affinage
4. Evaluer les éléments du bac d'affinage
5. Réordonner le bas d'affinage
6. Purger les bacs

# Planification d'un *Sprint*



- ❖ Confirmer les stories prêtes
  - ❖ Définition de prêt et fini
  - ❖ Evaluer la nombre de points d'une story
    - ❖ Ponts de récit ou journée idéale (homme/jour)
  - ❖ Organisation de l'essaimage (plusieurs stories en parallèle, répartition des ressources)
  - ❖ Décomposition des stories en tâches
  - ❖ Affectation des tâches aux développeurs





# Exécution d'un *sprint*



- ❖ Conception, réalisation et test des *stories*
- ❖ Organisation, affectation des tâches
- ❖ Inclut les *sprints* journaliers

# La mêlée quotidienne



- ❖ Courte : ~15mn
- ❖ Bilan: mise à jour du tableau de la story
  - ❖ Qu'est-ce que j'ai fait hier ?
  - ❖ Qu'est que je vais faire aujourd'hui ?
  - ❖ Quels sont les obstacles que j'ai rencontrés ?
- ❖ Objectif :
  - ❖ Rythmer le sprint (stories finies, prêtes)
  - ❖ Recenser les obstacles

- ❖ *Temps de la rétrospective*
  1. *Se remémorer*
  2. *Evoquer le positif*
  3. *Evoquer le négatif*
  4. *Proposer des axes d'amélioration*
  5. *Choisir les actions*

# Revue d'un *sprint*



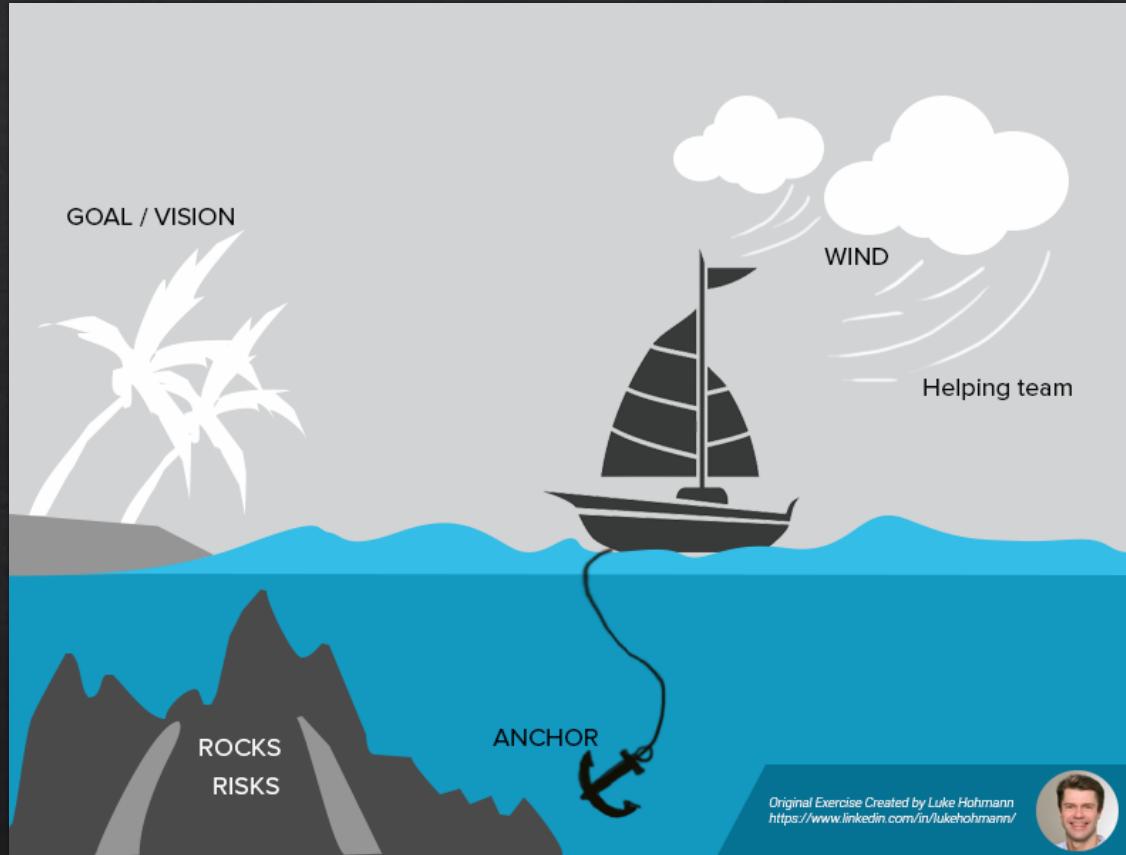
- ❖ Démonstration de chaque story finie
- ❖ Collecte du feedback
- ❖ Evaluation du niveau de réalisation de l'objectif
- ❖ Evaluation de l'impact du travail réalisé et décision d'une release (livraison) ou pas

# Rétrospective d'un sprint

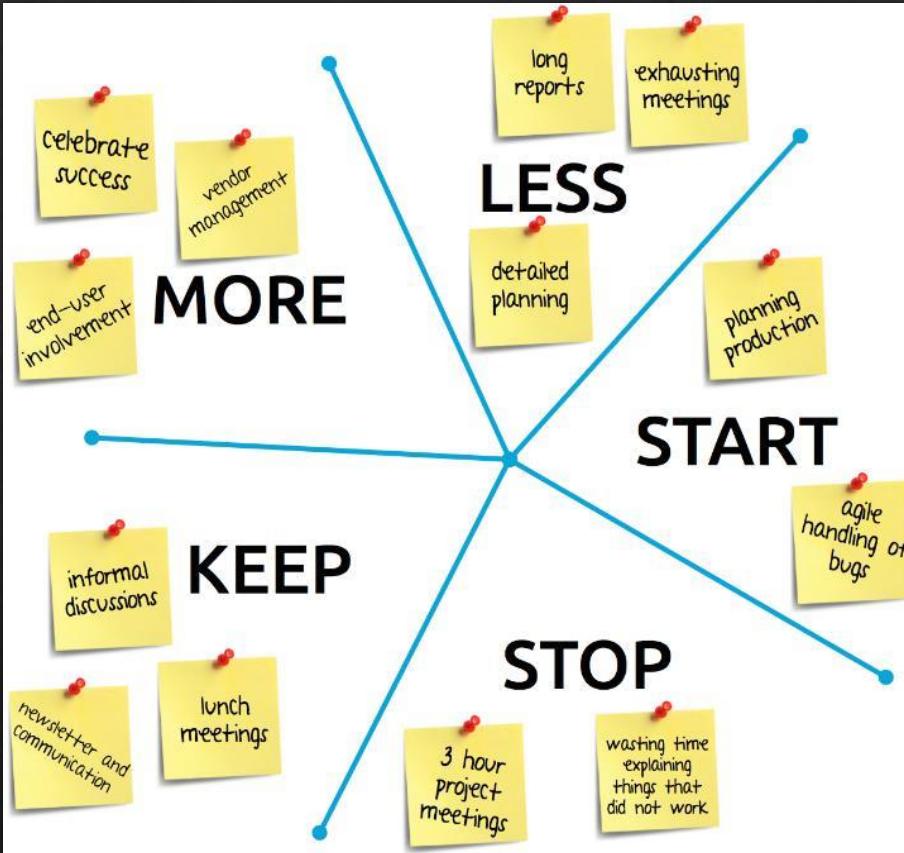


- ❖ Collecter les informations sur le sprint passé par rapport à la pratique SCRUM
  - ❖ Ce qui c'est bien passé, moins bien passé
- ❖ Identifier les choses à améliorer
- ❖ Décider d'améliorer certaines choses
  
- ❖ Combinée à la revue, de courte durée

# Rétrospective de sprint : « sailboat »



# Rétrospective de sprint : « Starfish »



# Solidification d'une *release*



- ❖ Tests de qualité de services (performances, coût, sécurité ...)
- ❖ Documentation
- ❖ ...

# Validation d'une release



- ❖ Test d'usage avec le client, des utilisateurs

# Outils

- ❖ Confluence
- ❖ Jira
- ❖ ...

# Problèmes de développement

- ❖ Dépassemement des délais
- ❖ Abandon
- ❖ Détérioration du système
- ❖ Défaillances
- ❖ Incompréhension
- ❖ Turnover (le développeur s'en va)
- ❖ Inutilité

