Cours 2
Développement agile avec Scrum

Priorisation, Estimation, Planification de Sprint

V. Deslandres © – IUT de LYON

Sommaire

Les **étapes et cérémonies** du développement agile avec SCRUM

Les user stories

(fin)

3 types de Story

- **User story (US)**
  - Besoin exprimé par le Product Owner (PO)
- **Story technique**
  - Ex.: tester le framework Guava pour la gestion du cache, tester l'algo de cryptage des mots de passe
  - Rarement exprimée par le PO ! Mais déduite d'une US
- **Default Story**
  - soit un défaut constaté
  - soit l'évolution d'une story initiale

Dans tous les cas : **simplicité et concision**

Notion de Fini (DoD)

- En Agile, le **DoD** est fondamental
  - ***Definition of Done***
- On va se mettre d'accord **AVANT** sur ce qu'est une US **terminée**
  - Quand les Test Unitaires passent
  - … en plus, la doc est faite
  - … en plus, la version anglaise est terminée

Avec l'Agile, on va éviter **le syndrôme des 90%**, où tout est commencé mais pas encore complètement terminé

Eviter la ***dette technique***

« Il *suffira* de faire appel à tel composant »

Tests d'Acceptation

- Définis par le PO, discutés
- Peuvent servir au TDD (*Test-Driven Development*)
- C'est là qu'on discute des **détails !**
  - Identifier les **conditions** qui permettent de dire que la tâche ou la US est terminée
- Soit pour la user story suivante :

> **En tant** que membre du site,
> **je peux** annuler ma réservation d'hôtel
> **Afin** de tenir compte d'imprévus dans mon voyage

Tests d'Acceptation : illustration

Product Owner

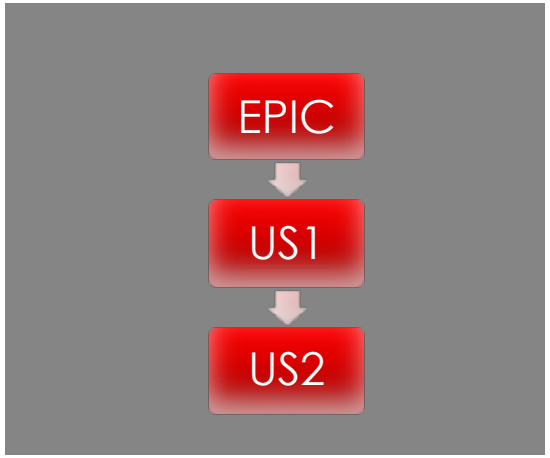
En tant que membre du site,
je peux annuler ma réservation d'hôtel
Afin de tenir compte d'imprévus dans mon voyage

- Vérifier qu'un email d'annulation est envoyé au voyageur et à l'hôtel
- Vérifier que si elle a lieu au moins 15 j. avant la date prévue, pas de charge imputée au voyageur
- Vérifier qu'un *premium* peut annuler n'importe quand sans charge suppl.
- Vérifier qu'un *non premium* paie 10% du montant de la résa si annulation moins de 15 j. avant la date prévue

Que penser de ce test d'acceptation ?

Manque de précision. Proposition :

Pour aller + loin : découper une EPIC en stories



- Si la story est trop grosse pour être développée en un seul sprint, ce n'est pas une story, mais une **EPIC**
  - Ex. EPIC : gestion d'un forum, gestion d'un agenda
  - Svt, les EPIC de priorité moindre ne sont pas même découpés en US

Techniques pour découper

- Par scénario
- Par opération CRUD
- Par type de données
- Par niveau de complexité, etc.

Rappel : une US tient sur un sprint, et au minium 4 user stories par sprint

LA CAPTURE des EXIGENCES

Product Owner

PRIORISER le BESOIN

Comment identifier ce qui est important ?

2 méthodes proposées : **KANO**, **MosCoW**

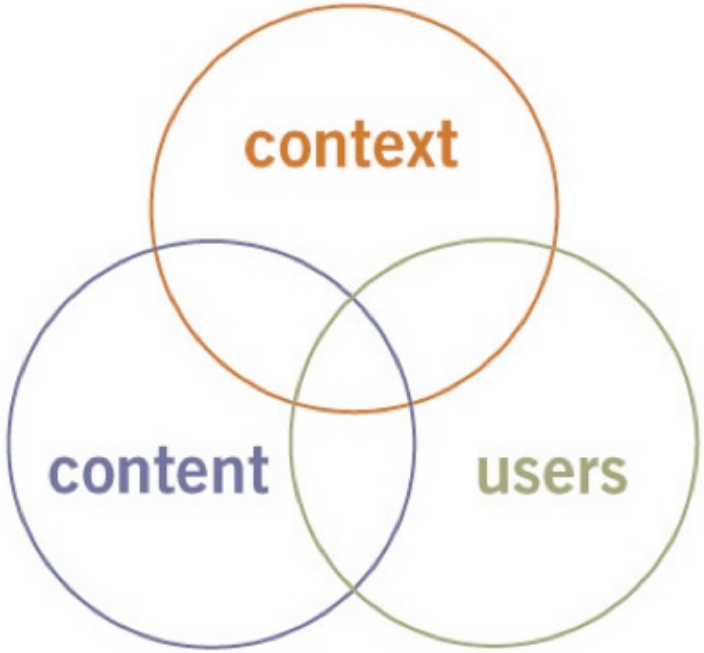
Affectation de la **valeur** Client

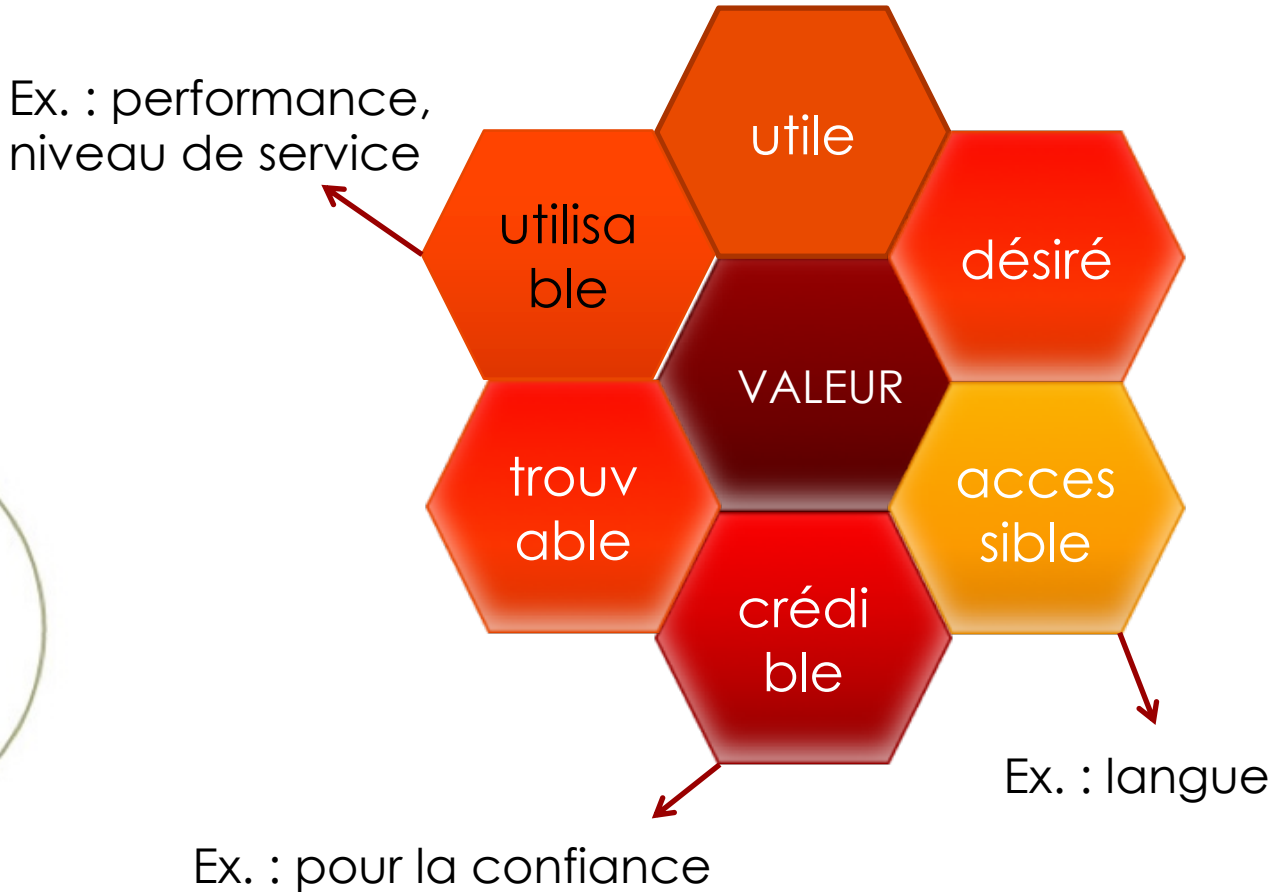
Objectif : PRIORISER les US

- A l'issue du **brainstorming** avec les utilisateurs, de nombreuses idées ont été avancées. L'objectif est alors de *réduire le nombre d'éléments* : les fédérer et les regrouper en thèmes communs.
- Comment trier le **backlog Produit** en fonction de ce qui est le plus important pour les utilisateurs et pour l'organisation ?

- **Valeur Métier** (business value)
  - C'est la partie BUT (« afin de ») des US
- Priorisation effectuée par le **PO + utilisateurs** en Scrum
  - (en XP, effectuée par le Client)

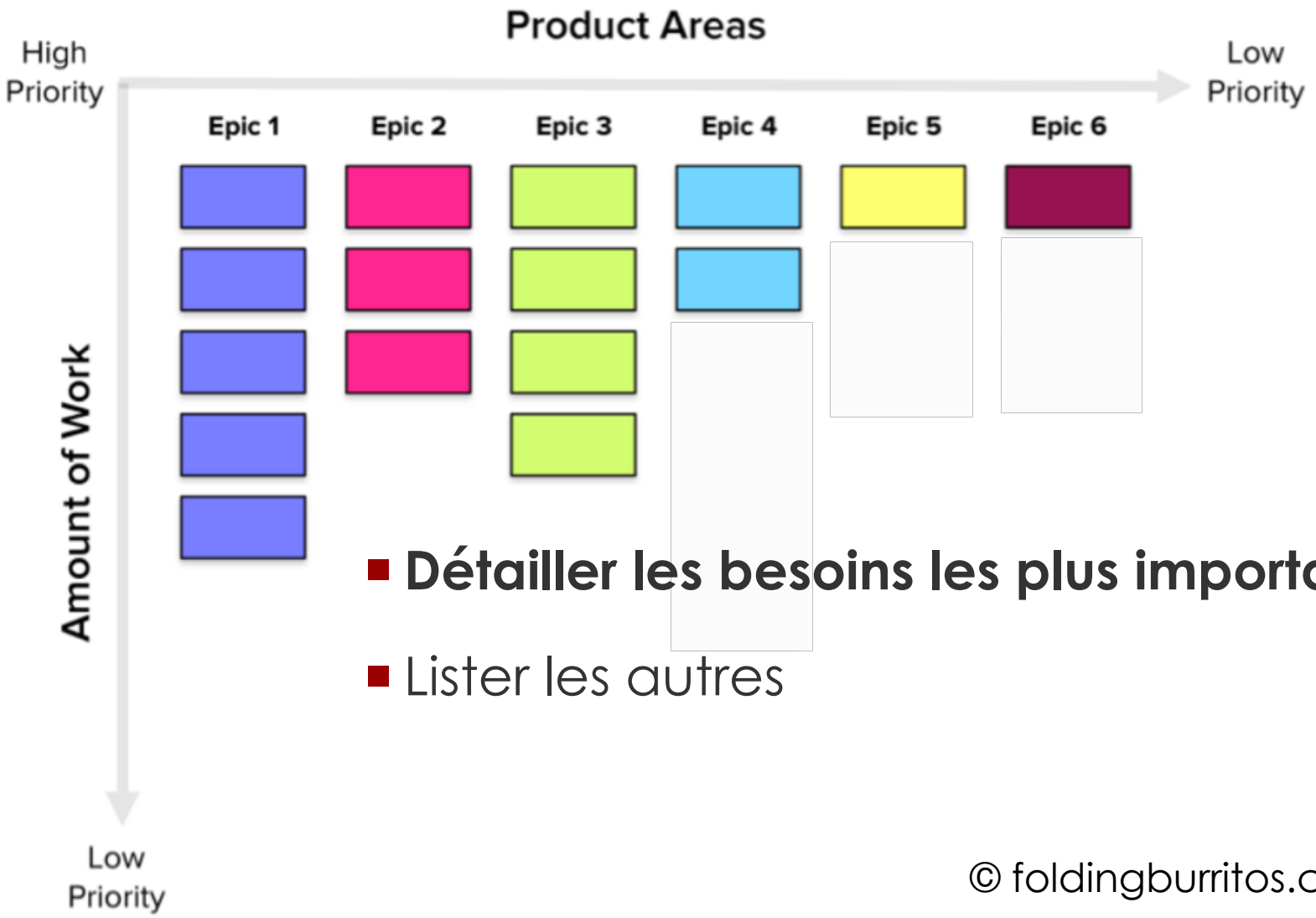
Qu'est-ce qui définit la valeur Métier ? Pas facile !





Agile UX - User Experience - *Peter Morville*

Ce que l'on souhaite obtenir :



- **Détailler les besoins les plus importants**
- Lister les autres

© foldingburritos.com

Prioriser le besoin pour le **Métier** : 2 méthodes

- **Matrice de Kano**
  - Méthode aussi utilisée en Marketing pour comprendre l'attitude des gens à l'égard d'une offre
- **MoSCoW**

  Must / Should / Could / Won't
  - Méthode la plus facile / rapide
  - (Il en existe d'autres : story mapping)

1- Matrice de Kano

3 types de fonctionnalités

- **Obligatoires**
- **Linéaires** : ajoutent un vrai plus au produit; sans elles, le produit ne sera pas aussi bon aux yeux des utilisateurs
- **Attractives** : si présentes, les clients adorent; pas d'impact si absentes, ils n'y pensent pas

Définies selon 2 axes :

présence / satisfaction

Attention : évalue la perception future des fonctionnalités **sur le client,** pas la satisfaction réelle

Matrice de KANO

Fonctionnalités **seuil ou essentielles**
Ex.: syst freinage véhicule

Fonctionnalités **linéaires**
Ex.: véhicule économe en carburant, performances moteur

Fonctionnalités **attractive**
Non indispensable, bonne surprise
Ex.: qualité syst audio d'un véhicule

Illustration : café

- Vous entrez dans un bar et demandez un **expresso**

- Donner des illustrations pour :
  - Fonctionnalités seuil
  - (Insatisfaction)
  - Linéaires
  - Heureuse surprise

À quoi ça sert d'imaginer les causes d'insatisfaction ?

Illustration : café
par exemple

- **Fonctionnalités essentielles**
  - Bonne taille d'expresso
  - Mousseux
  - Soucoupe
  - Cuillère
- **Insatisfaction forte**
  - Longue attente
  - Café froid
  - Soucoupe sale
  - Pas de sucre (même si on n'en prend pas)
- **Fonctionnalités linéaires**
  - Durée de l'attente
  - Empathie du serveur-euse
  - Qualité du café
- **Heureuse surprise**
  - Un verre d'eau
  - Un gâteau ou un chocolat

http://christian.hohmann.free.fr/index.php/six-sigma/les-outils-de-la-qualite/476-le-diagramme-de-kano

Questionnaire de Kano : 2 parties

- Niveau de satisfaction des fonctionnalités **opérationnelles** (le fonctionnel). Avis de *n* utilisateurs.

Pour un produit 'Appareil Photo' (ici 9 personnes ont trouvé qu'avoir un viseur était le min)

Si cet item est disponible :	1-Ça me fait plaisir	2- C'est le minimum	3- Ça m'est égal	4-Je l'accepte	5- Ça me dérange
Viseur	**4**	**9**		**2**	

- Ressenti des fonctionnalités **absentes** (le dysfonctionnel)

Si cet item est absent :	1-Ça me fait plaisir	2-C'est le minimum	3- Ça m'est égal	4- Je l'accepte	5- Ça me dérange
Viseur			**4**	**3**	**8**

- **Ce double questionnement est la clef de la méthode**
  - Car ***satisfaction*** et ***insatisfaction*** ne sont pas ***symétriquement opposées***

Kano : grille

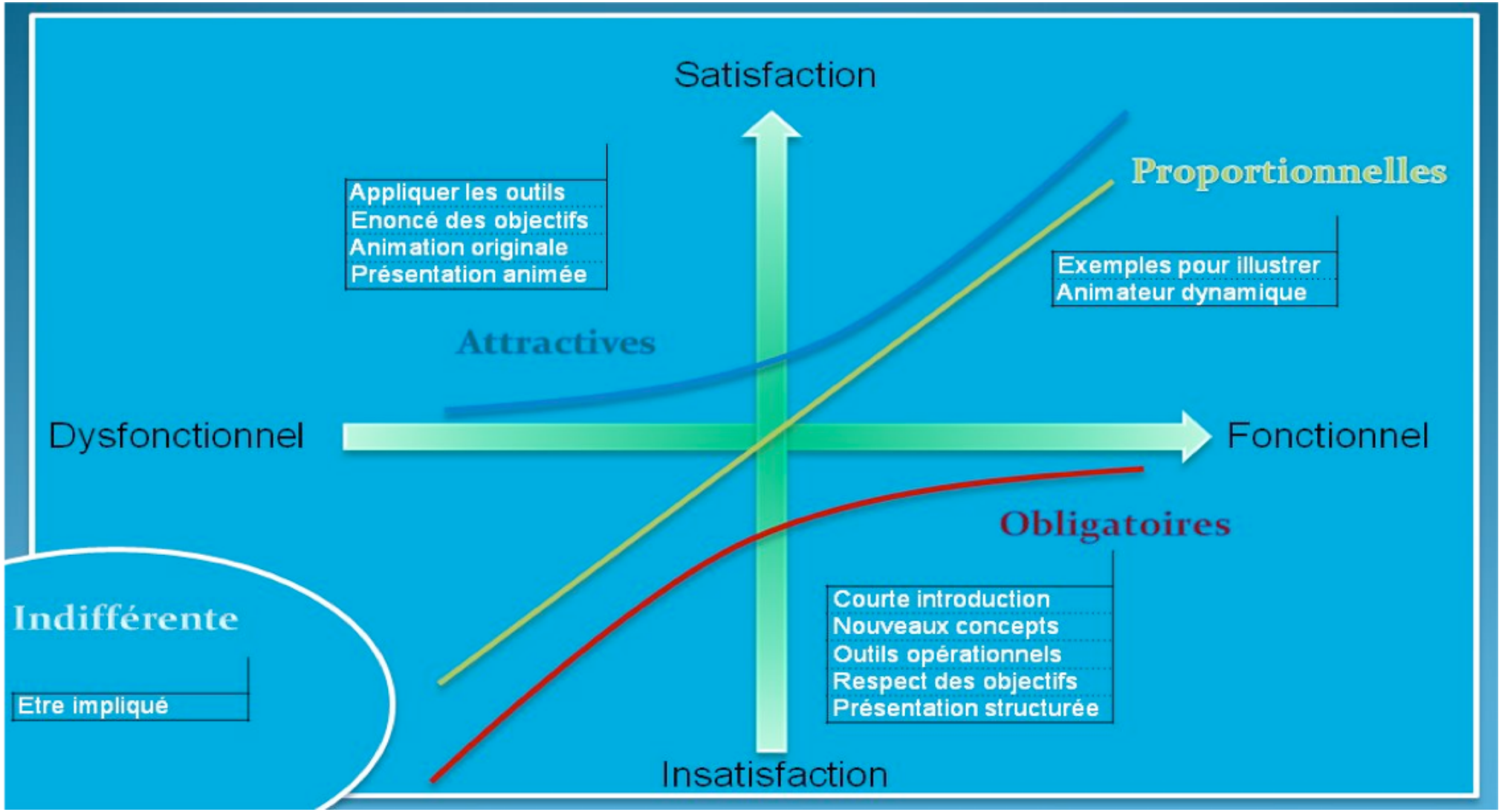
- Permet d'identifier les catégories de fonctionnalités parmi :
  - **O**bligatoires
  - **A**ttractives
  - **P**erformantes
  - **I**ndifférentes
  - **C**ontraires (hostiles)
  - ***D****éfaut (erreur)*

		Question dysfonctionnelle				
		1	2	3	4	5
Question fonctionnelle	1	D	A	A	A	P
	2	C	I	I	I	O
	3	C	I	I	I	O
	4	C	I	I	I	O
	5	C	C	C	C	D

1: Ça fait plaisir
5: Ça dérange

CRITIQUES de KANO
- Long et fastidieux
- Utilisé en Agile pour prioriser qd on a bcp d'utilisateurs

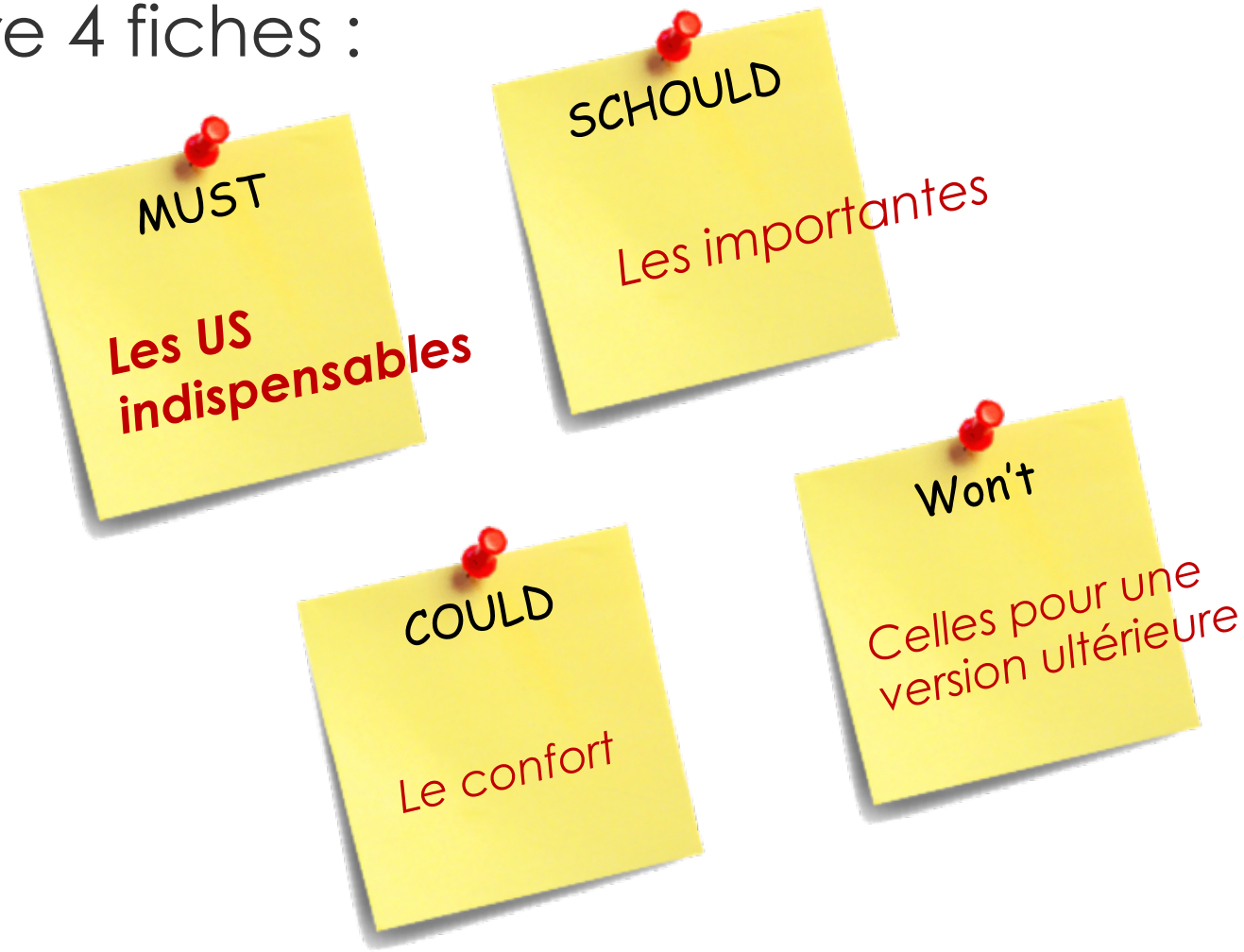
Un exemple KANO pour une formation



http://florentfouque.blogspot.fr/2008/10/define-la-matrice-de-kano.html

2- Prioriser avec MoSCoW

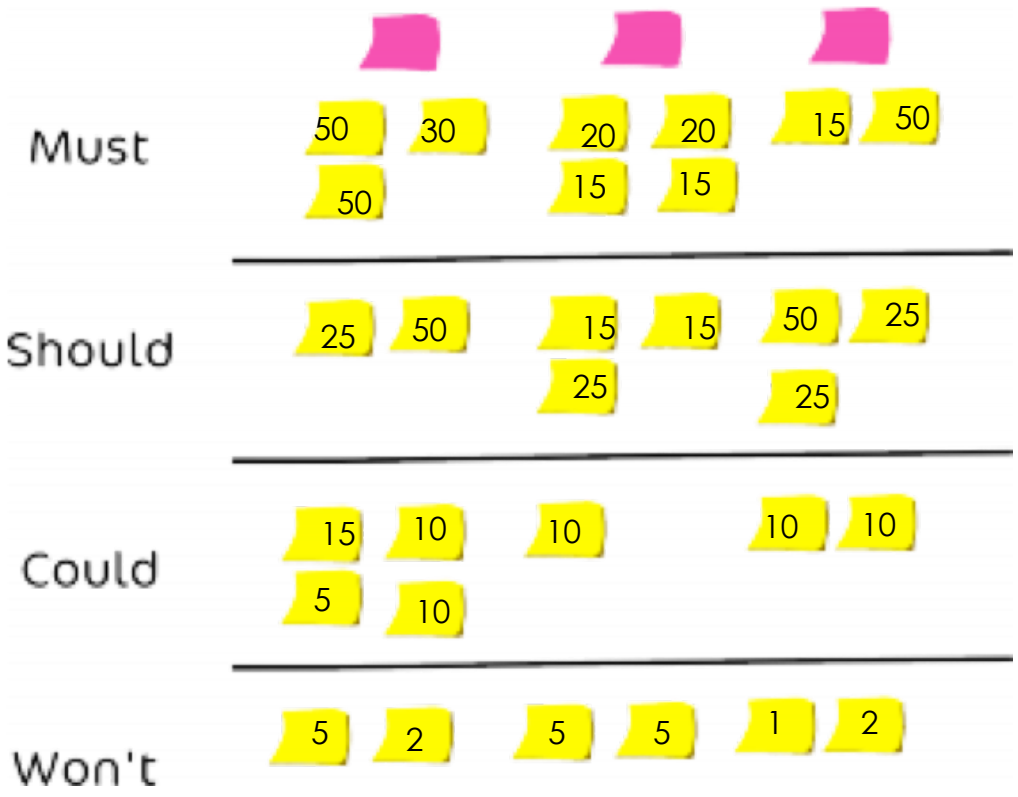
- On prépare 4 fiches :



- En équipe, on place les US sur les fiches
  - Soit l'un après l'autre, soit par vote : argumentation, débat
- Max **8** par fiche
- REGLE : tous les items MUST devront être finis **sur les 2 premiers** sprints

Quantifier la **valeur** Client

- Une fois **la liste priorisée des user stories**, on attribue une valeur globale au projet (par ex. 100 ou 500 points) et on répartit les valeurs sur les US
- En restant **cohérent** avec les priorités



265

230

70

On inscrit la valeur dans un angle de la User Story

LA PRIORISATION: Je retiens….

La somme des points Valeur définit la *business value* du projet

- Il s'agit de regrouper/trier les items du backlog selon la **valeur Métier**, avec les utilisateurs ou le client
- On hiérarchisera le *Backlog* en fonction de la VALEUR et de l'EFFORT (étape suivante)
- Définir la valeur Métier n'est pas évident (notion d'utilité, de désir, de facilité et fréquence d'utilisation etc.)

2 méthodes pour PRIORISER le BESOIN

- Il existe différentes méthodes pour trier les US en fonction de la valeur Métier, ici on a vu :
  - La **méthode de KANO** (double questionnaire selon la présence/absence d'une fonctionnalité) : identifie les fonctionnalités Obligatoires, Attractives et Linéaires
  - **MoSCWo** (Must – Should – Could – Won't)

POUR EN SAVOIR PLUS…

Un exemple de priorisation Agile avec la matrice de KANO

https://blog.myagilepartner.fr/index.php/2017/01/09/quest-ce-que-le-modele-de-kano/

LA CAPTURE des EXIGENCES

Dév., testeurs, designers

ESTIMER
Définir l'effort nécessaire

Estimer le coût de réalisation

Estimer l'effort

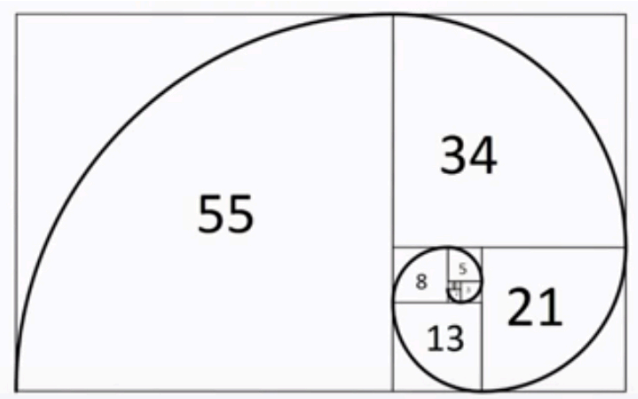
- Une fois les US affectées de valeur Métier avec le PO, l'équipe va **quantifier l'effort nécessaire pour les réaliser**

2 méthodes d'estimation

- Planning POKER
- T-Shirt Size

1- Estimer Effort avec le Planning Poker

- Fonctionne avec le système de points de Fibonacci :
  - **1(facile), 2, 3, 5, 8, … (difficile)**
  - chacun dispose des cartes de Fibonacci



- Fonctionnement
  - Le tas des US est au centre, on en retourne une
  - Chaque Dév retourne la valeur d'effort qu'il estime juste, tous en même temps
  - Si différences, on discute des valeurs extrêmes
  - On se met d'accord sur un effort

Désaccord sur l'estimation ?

- En cas de profond désaccord :
  - Discuter ! Origine du problème ?
  - soit incertitude sur le **produit**, la fonctionnalité
  - Soit sur les **technologies**
- Dans les 2 cas -> **une solution**
  - Mettre la User Story de côté, creuser les choses pour diminuer l'incertitude, et la ré estimer lors de la prochaine séance

(on peut aussi prendre la valeur moyenne)

2- Estimer l'effort avec la méthode T-SHIRT size

- Certains préfèrent attribuer un **niveau de difficulté** aux US et tâches plutôt que lui donner une valeur d'effort
  - Plus souple
  - Plus facile de comparer un tâche par rapport à une autre
- Méthode du T-Shirt Size : on choisit les catégories

  XS, S, M, L, XL
  - Soit définir une référence : M = 2 jours
  - Soit choisir une **User Story étalon**, connue et maîtrisée, et lui affecter l'effort bien
    - *Ex. : développer l'interface de saisie de commande Client estimé S*
  - Mesure de **l'effort « relatif »**
    - *l'interface de saisie des infos Client = 2x plus long, disons M*

Tshirt-size Méthode

- Les tâches **XL** sont en général **floues**
  ➔ Les découper et repositionner
- L'objectif est de trier les *items* en complexité

- On attribue ensuite des **points** de Fibonacci aux catégories
  - Niveau difficulté faible → effort de développement faible
  - Par ex. :

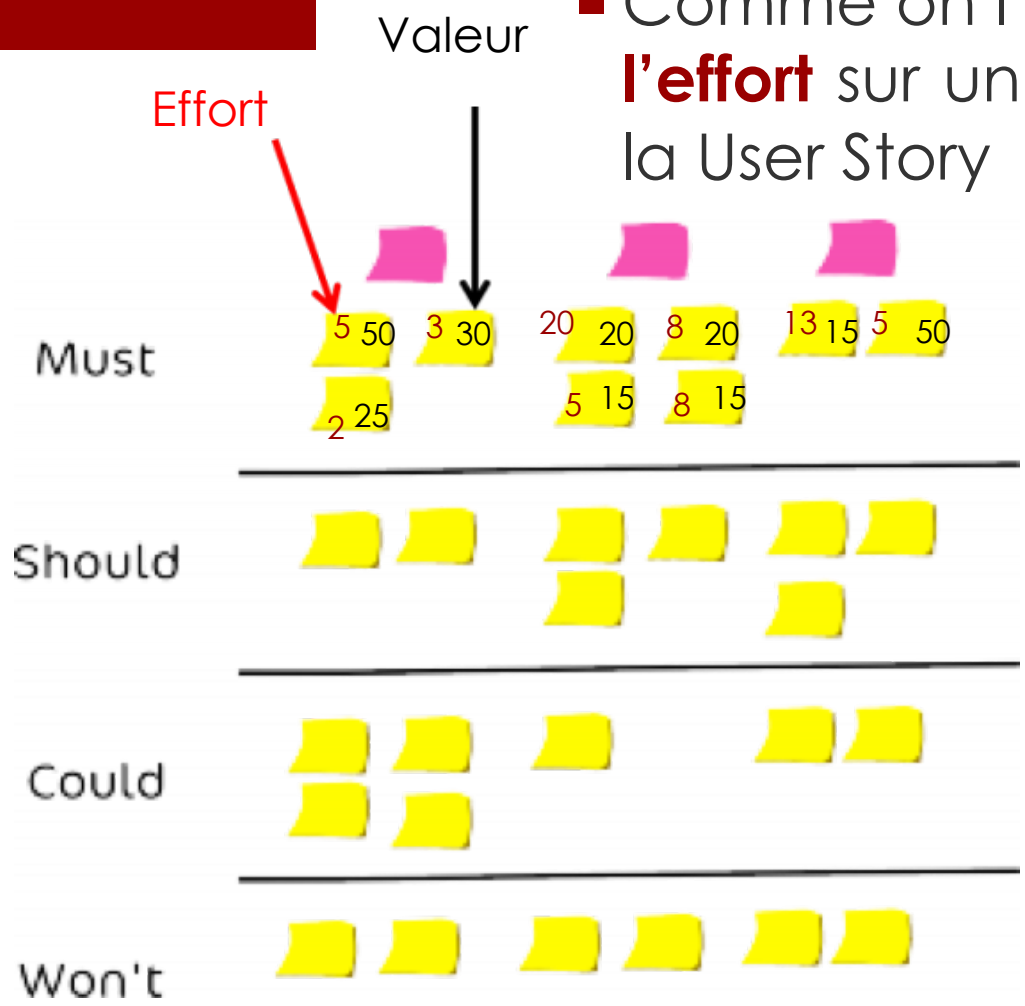
XS : 0.5 S : 1 **M : 5** L : 8 XL : 13

2 principes pour l'Estimation : Faible précision / Marge collective

- **Faible précision**
  - Rester flou, inutile d'aller trop dans le détail de l'estimation
  - Ça serait du temps perdu ! Car les estimations sont tjrs fausses (Pablo Pernot). Ce qui compte c'est l'estimation **relative**.
- **Marge collective**
  - Chaque développeur doit être **honnête** dans son estimation
  - Parfois, quand on estime une US, on se donne « un peu de mou » pour gérer les imprévus
    - Chacun **surévalue** ainsi son estimation
  - Or c'est au bénéfice de **l'équipe** qu'il faut donner de la marge
  - Il y **a moins de gâchis** en gérant une marge globale au **niveau de l'équipe**, que par personne

Compléter les US

- Comme on l'a fait pour la VALEUR METIER, on inscrit **l'effort** sur un autre angle de la carte représentant la User Story



Valeur Métier vs. Vélocité

- La **somme des points d'effort** définit l'effort global nécessaire pour développer le produit
  - Il est différent de la *Business Value* (somme des points de Valeur Métier)
  - Il fait référence à la **productivité** de l'équipe
- En comptant **combien de points d'efforts sont réalisés par itération**, on a une idée de la capacité à produire du code de l'équipe
  - En Scrum on parle de ***vélocité*** de l'équipe
  - Souvent **variable** en début de projet, généralement après qques itérations, la vélocité **se stabilise**

Un ex. de Backlog Produit

Backlog item	Acceptance Criteria	Effort	Value
US1 - En tant qu'internaute, je peux réserver l'hôtel en ligne	• Un email de confirmation est envoyé • Réservation doit être faite au min 24h avant date	8	**25**
US3 - Améliorer la gestion des exceptions	Pas de msg "Exception …" même en cas de pb	21	**10**
US4 - En tant que membre, je peux modifier les dates d'une réservation	…	8	**20**
US6 - En tant que membre, je peux annuler une réservation	• Un email de confirmation est envoyé • Ne peut être annulée qu'au moins 15 jours avant la date	11	**25**
US7 - En tant que Resp. Hotel, je peux voir les réservations à venir	…	21	**15**

Hiérarchisation du Backlog & Planifications

Release : lot

Sprint : itération

Planifier... Ça signifie quoi ?

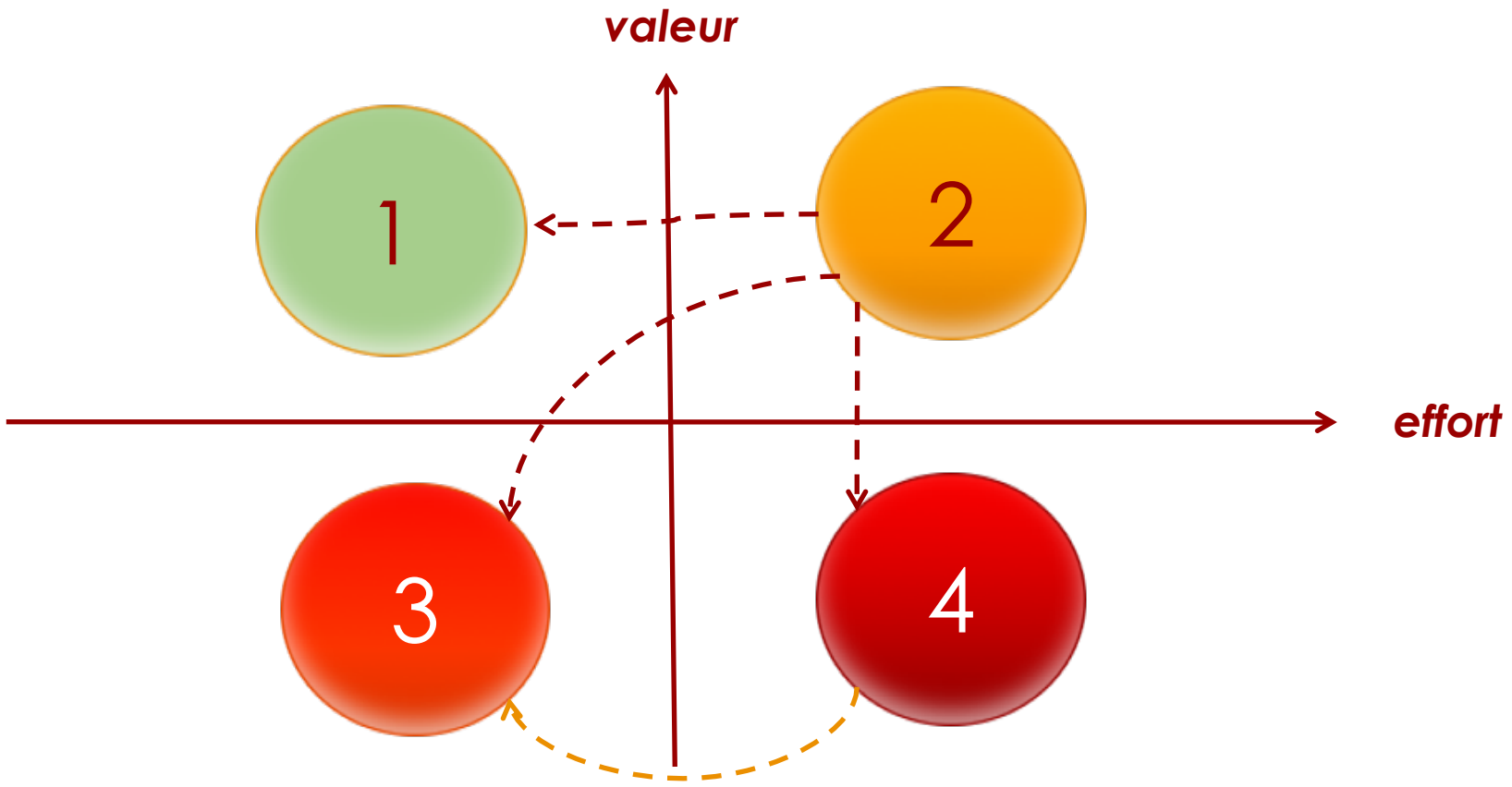
- Il y a 2 planifications en SCRUM :
  - La **planification de la séquence des sprints**
    - Une fois le backlog défini
  - La **planification de sprint** (en début d'itération, découpage en tâches)
- Si le projet est important, on planifie aussi les **différentes versions du produit livrées** (releases)

Planifier ➔ hiérarchiser

- Planifier la **séquence des itérations :** ce qu'on va faire à chaque sprint
  - Quelles US vont être réalisées dans quel sprint
- Pour cela, il faut **hiérarchiser le Backlog Produit**
  - Non pas en fonction de la VALEUR métier seule
  - Mais du couple (valeur, effort)
- Objectif : faire ce qui est **le plus important** et **le plus facile** d'abord
- 2 méthodes pour hiérarchiser le backlog:
  - **Matrice Valeur / Effort**
  - **ROI**

1- Hiérarchisation du Backlog avec la **Matrice Valeur / Effort**

- Méthode la plus utilisée, la plus efficace



2- Hiérarchiser le Backlog Produit **avec calcul de ROI**

- Pour chaque User Story, on calcule le ratio :

Valeur Business de la US / Effort estimé

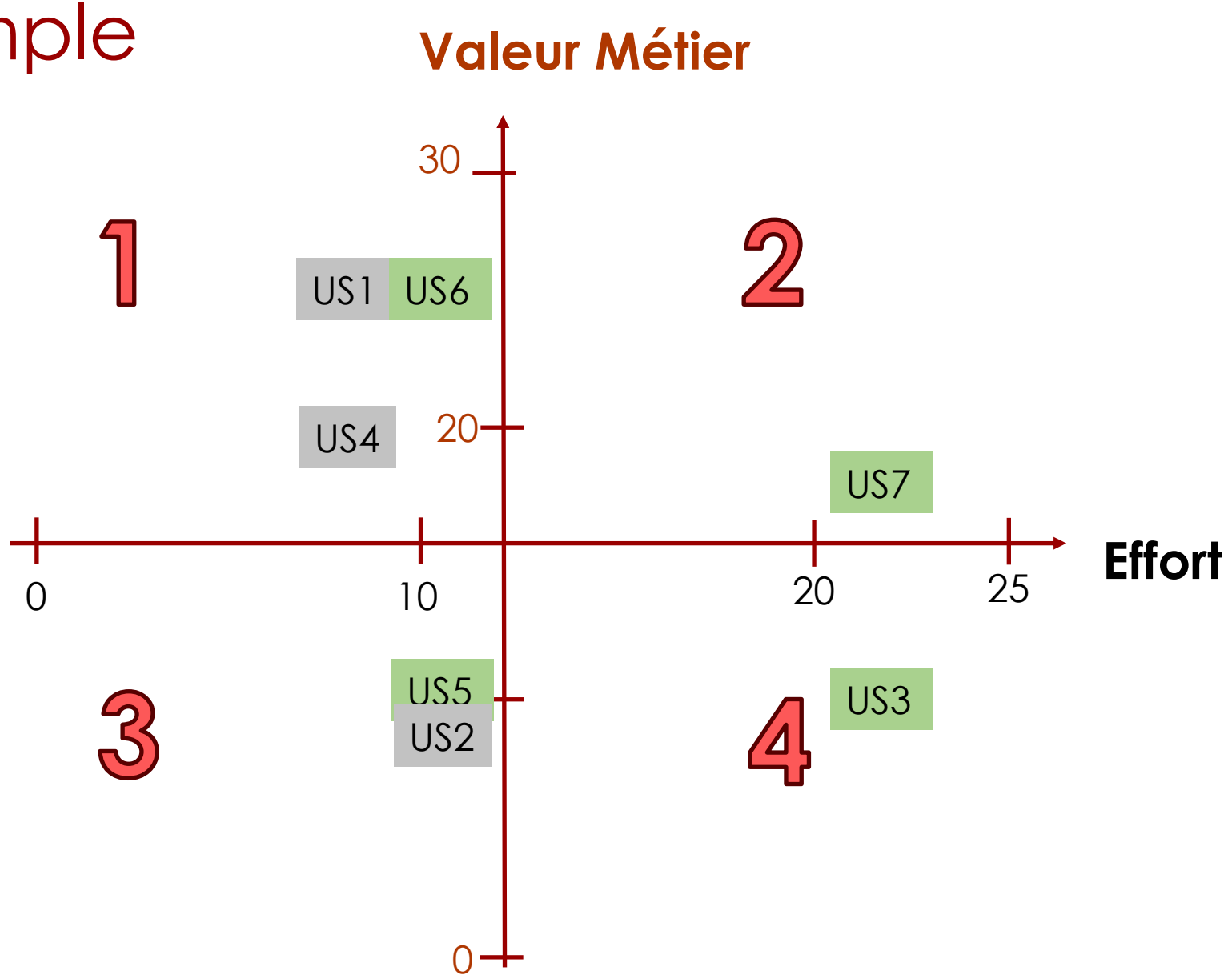
C'est le **ROI** (Return On Investment)

- Pour le premier sprint, on choisit les US qui ont le ROI le plus élevé

Avec les 2 méthodes, on obtient un **backlog de produit HIÉRARCHISÉ**

- On sait par quoi on va commencer à développer

Sur l'exemple



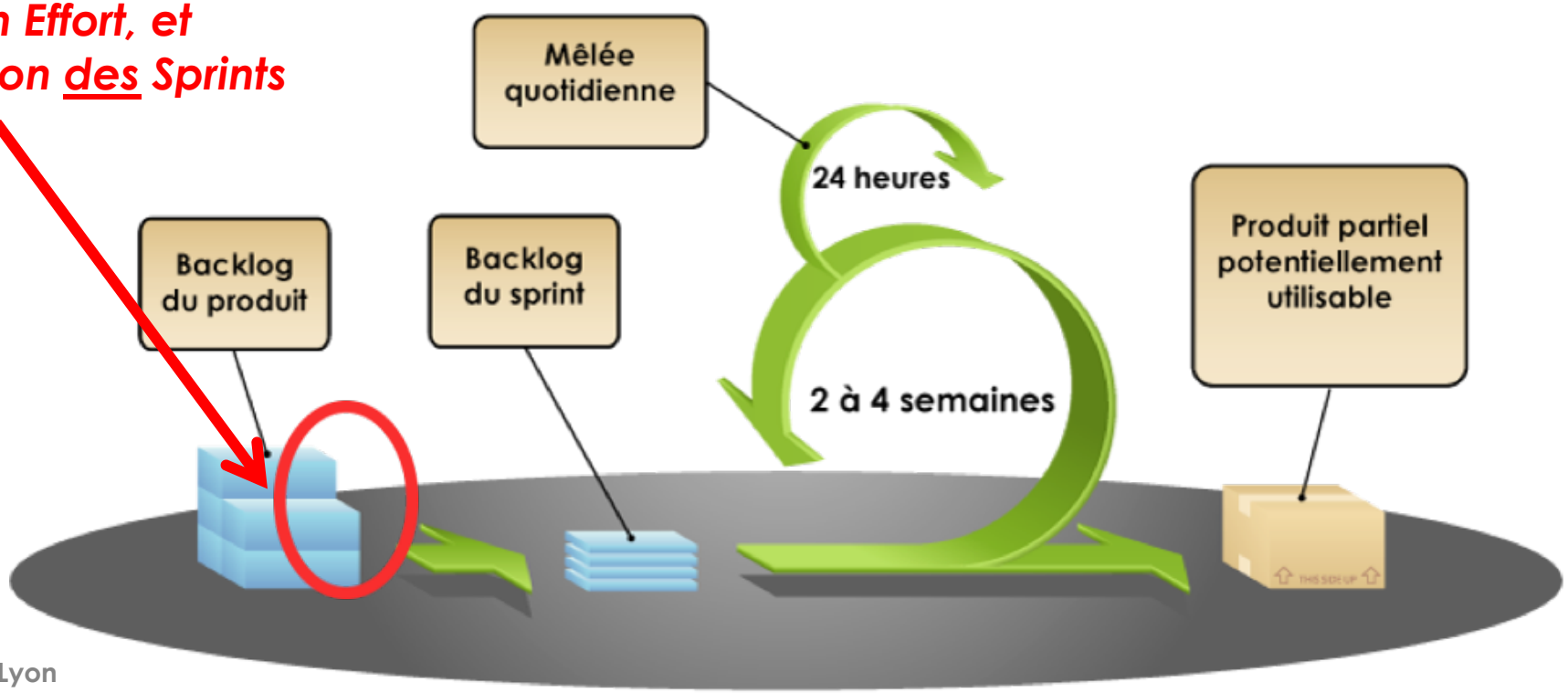
→ Backlog produit hiérarchisé

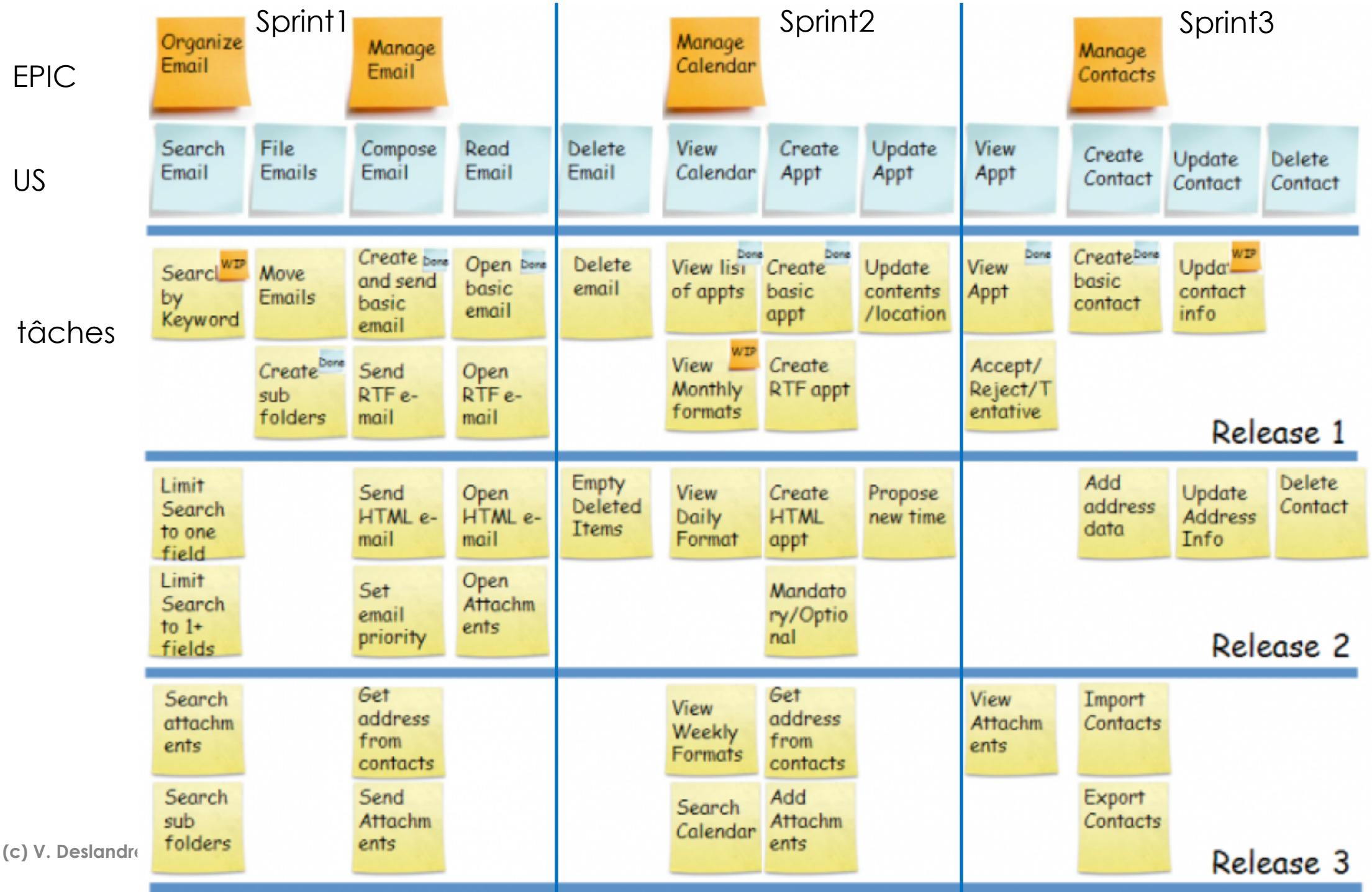
Priority	Backlog item	Acceptance Criteria	Estimate	Value
1	US1 - En tant qu'internaute, je peux réserver l'hôtel en ligne	• Un email de confirmation est envoyé • Réservation doit être faite au min 24h avant date	8	25
2	US6 - En tant que membre, je peux annuler une réservation	• Un email de confirmation est envoyé • Ne peut être annulée qu'au moins 15 jours avant la date	11	25
3	US4 - En tant que membre, je peux modifier les dates d'une réservation	…	8	20
4 …	US7 - En tant que Resp. Hotel, je peux voir les réservations à venir	…	21	15
7	US3 - Améliorer la gestion des exceptions	Pas de msg "Exception …" même en cas de pb	21	10

Planification SCRUM : planification des Releases & des Sprints

- Quand ? Une fois le Backlog hiérarchisé
- En général : 4 à 5 User Stories par sprint

Priorisation Valeur + estimation Effort, et Planification des Sprints

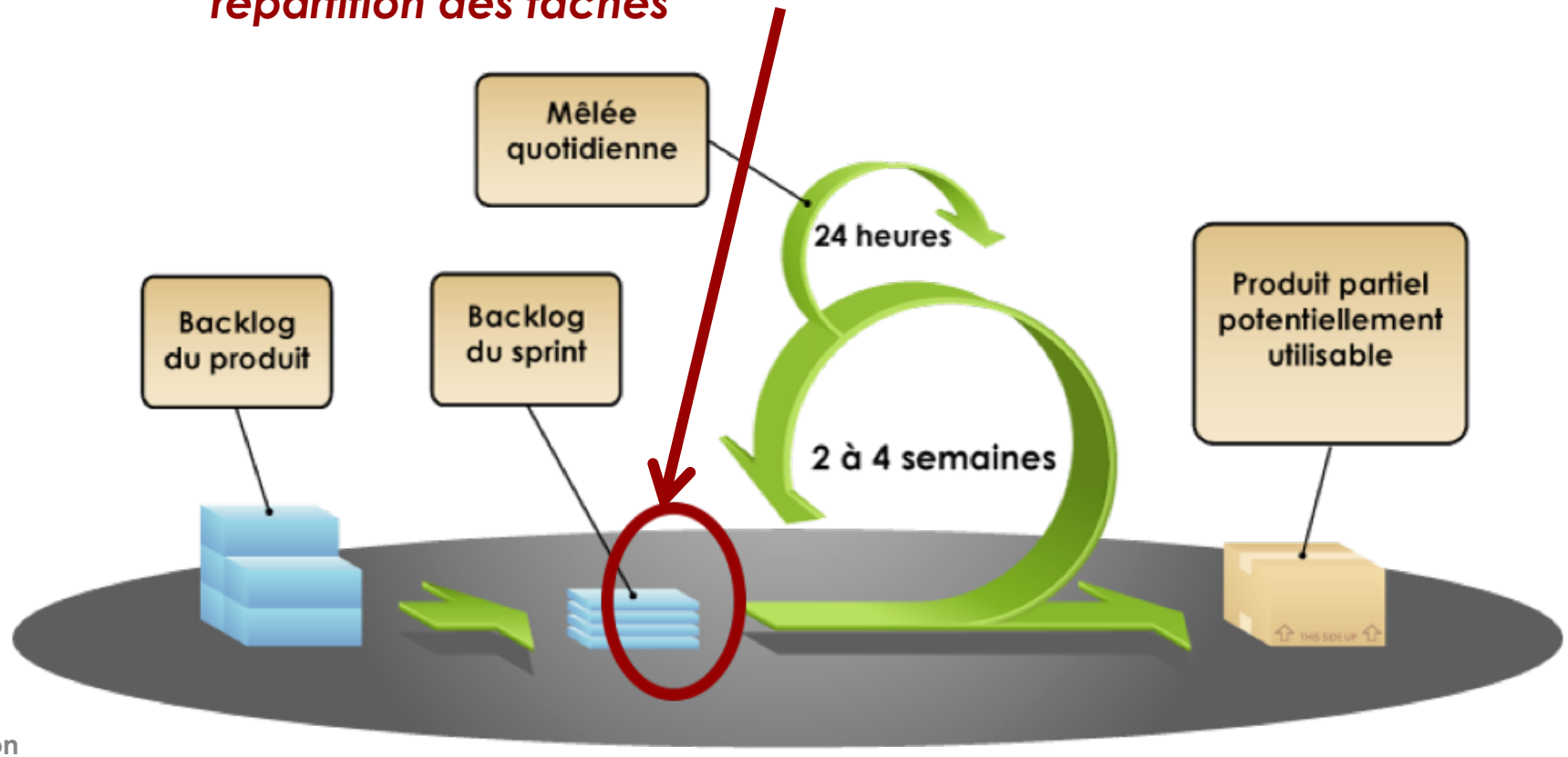




2ème Planification SCRUM : la planification de Sprint

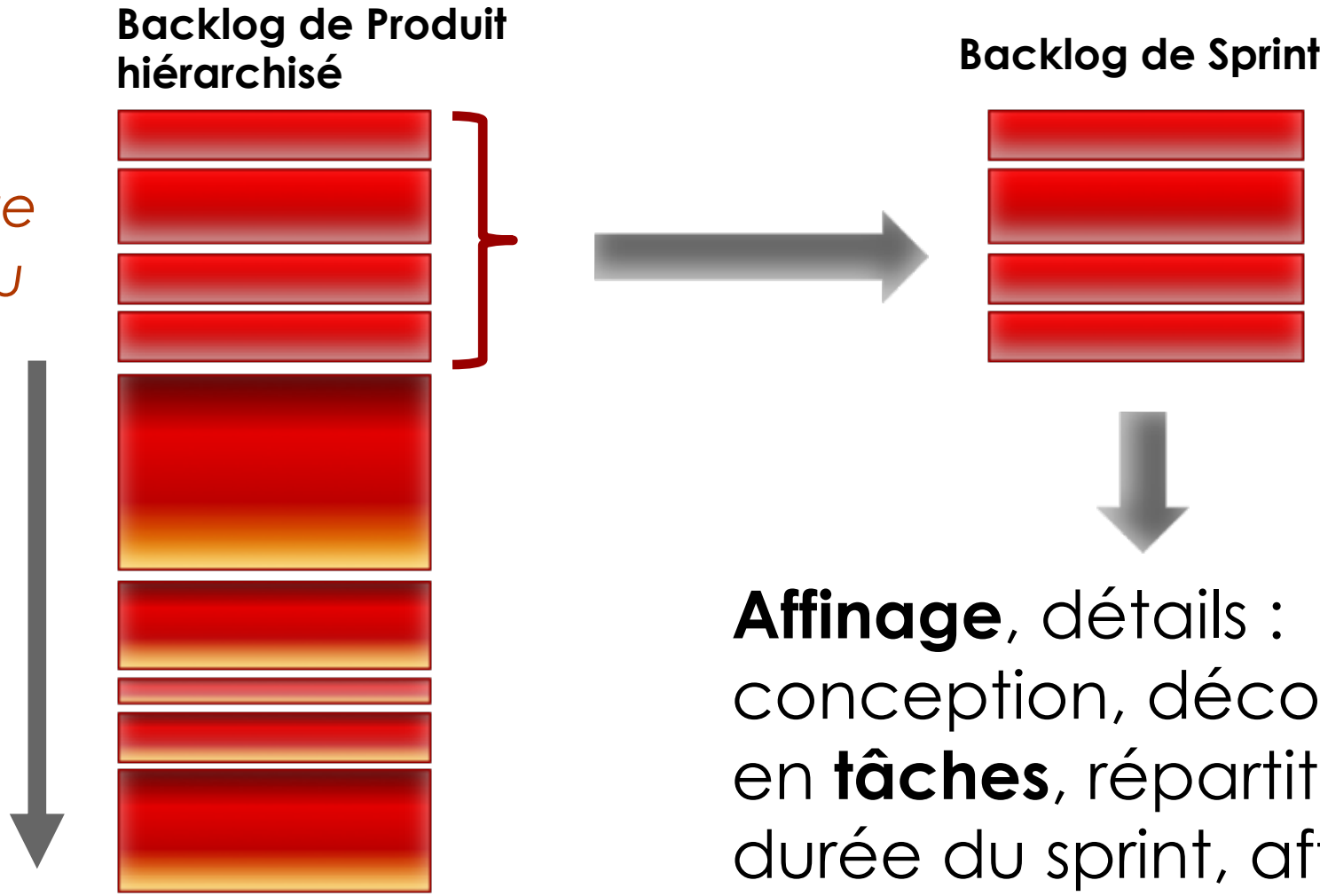
- Quand ? En début de sprint
- Durée : max 1h pour un sprint de 1 semaine (2h pour 2 semaines)

Planification de Sprint : découpage des US en tâches, répartition des tâches



Planification de Sprint

Priorité décroissante (fonction du couple : valeur et effort)



Affinage, détails : conception, découpage des US en **tâches**, répartition sur la durée du sprint, affectation aux développeurs

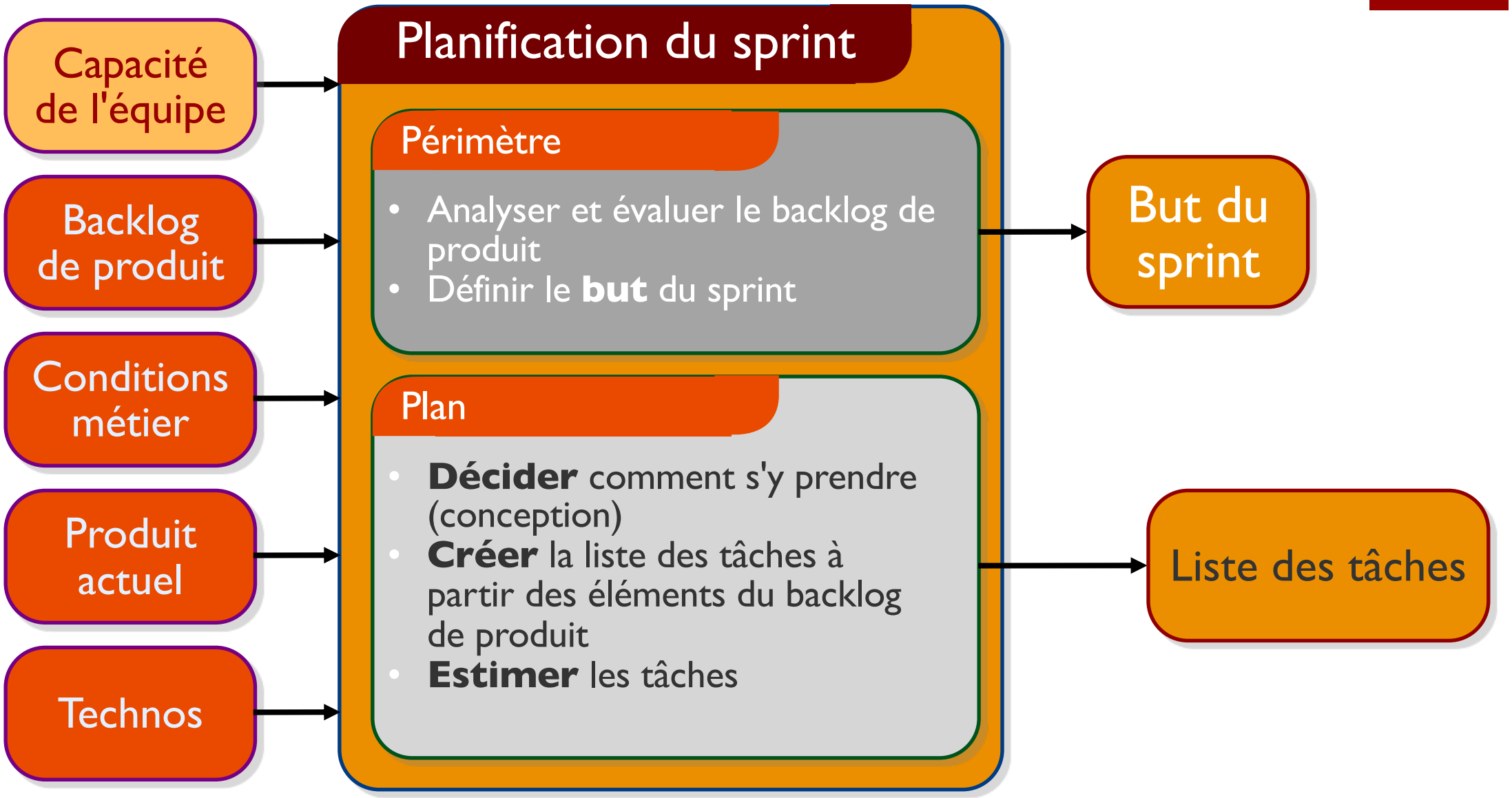
Planification de sprint : un exemple

- Les tâches sont identifiées et **estimées**
  - (parfois en heures comme ici)

- L'équipe définit une **somme de points d'effort** qu'elle est capable de fournir pendant la durée d'un Sprint (la **capacité**)
  - Pour le 1er sprint, on mise sur une capacité « a priori »
  - Elle sera ajustée au 2ème sprint

ETQ touriste dans la région, je veux voir les photos des hôtels afin de savoir si je vais m'y rendre

- Modéliser et implémener la couche de persistance (8h)
- Coder l'IHM (4h)
- Ecrire les tests (4h)
- Coder la classe *foo* (3h), etc.


Capacité de l'équipe
Backlog de produit
Conditions métier
Produit actuel
Technos
Planification du sprint
Périmètre
Analyser et évaluer le backlog de produit
Définir le but du sprint
Plan
Décider comment s'y prendre (conception)
Créer la liste des tâches à partir des éléments du backlog de produit
Estimer les tâches
But du sprint
Liste des tâches

Au boulot !

- **Prioriser le besoin** avec les valeurs Métier
  - Méthode KANO ou MosCoW
- **Estimer l'effort de réalisation** nécessaire
  - Méthode Planning Poker ou T-Shirt Size
- **Hiérarchiser le backlog**
  - Matrice Valeur/effort ou calcul du ROI
- **Planifier les sprints**
  - Répartir les US sur les itérations
  - (on aura 7 itérations)

Pour la prochaine séance: m'envoyer le backlog v2 avec toutes ces infos

Priorisation / Estimation : Je retiens

- On peut **prioriser** le besoin (les User Stories) avec 2 méthodes : Kano et MosCow
- **Estimer** l'effort avec 2 méthodes : Planning Poker et Tshirt Size
- **Hiérarchiser** le backlog avec la Matrice Valeur/ Effort (ou le ROI)
- **Planifier les sprints** et les releases
- Faire la **planification de Sprint** en découpant les US en tâches dont on évalue aussi l'effort