Module Agilité – Scrum Master 🡪 AGILE

# Introduction :

Présentation de Agile qui n’est pas UNE méthode de gestion de projet et qui n’a même rien a voir, nous allons voir egalement le Manifeste Agile. Puis présentation de Scrum qui est une méthode Agile.

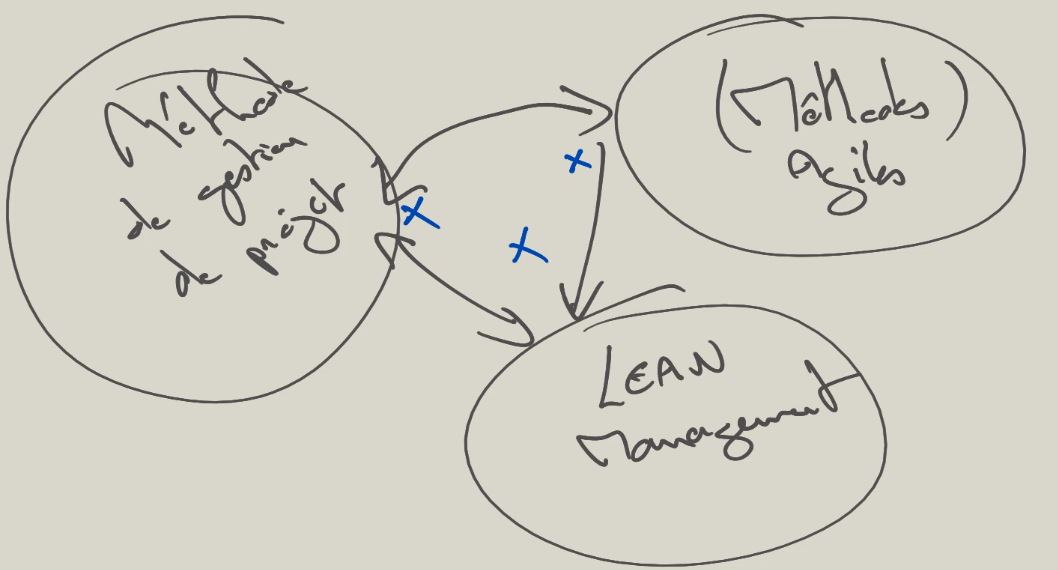
QCM Scrum, puis Expression de besoin, planification et estimation, Extreme Programming, DevOps,Kanban (Lean Management)

# Méthode Agile et La méthode de gestion de projet :

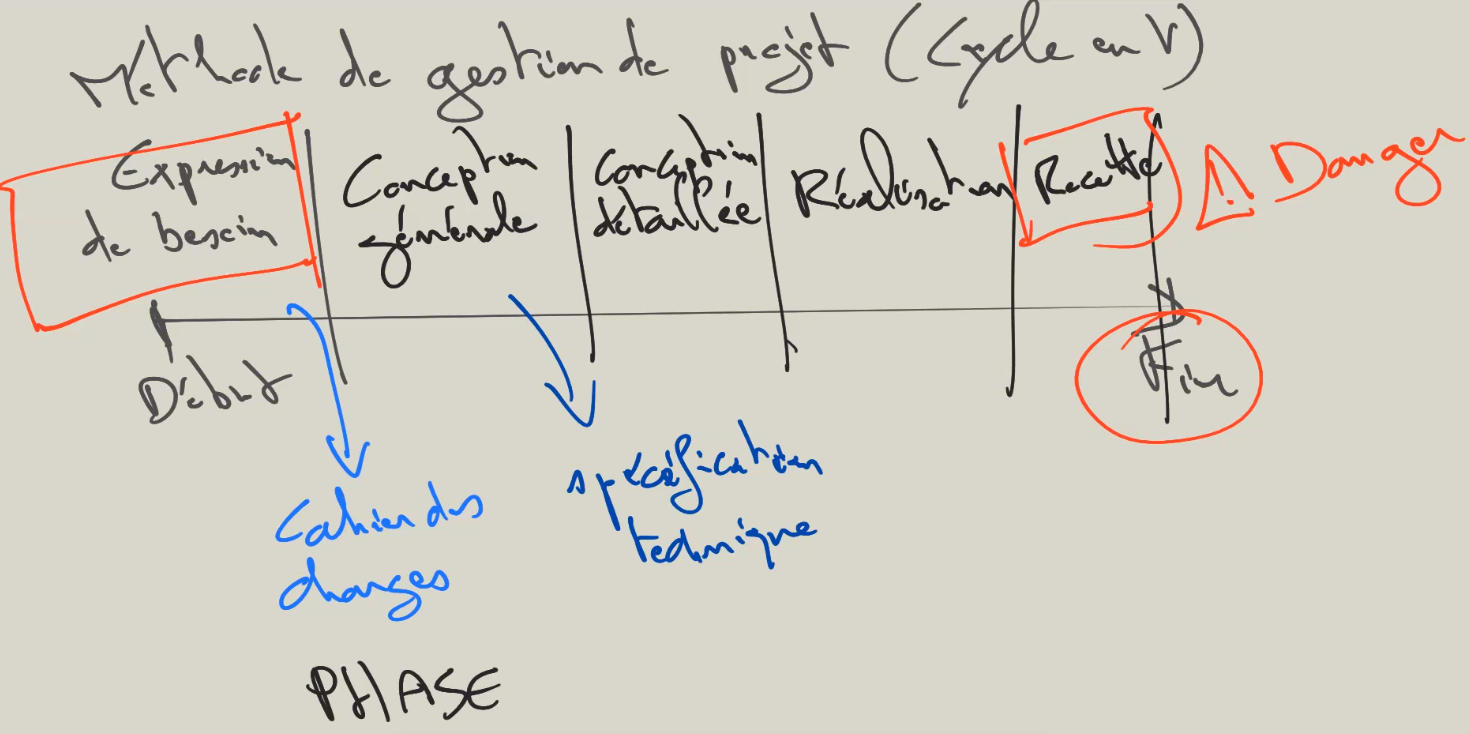
## La Méthode de gestion de projet = le cycle en V

Initialisé par des informaticiens qui travaillait sur le travail, c’est-à-dire qu’ils réfléchissaient sur leur façon de travailler qui étaient a la base plutôt en cycle en V.

Sachant une chose : la méthode du cycle en V **n’existe pas en de gestion de projet** en fait **LA méthode de gestion de projet C’EST le cycle en V** tout du moins une représentation graphique du cycle de vie d’un projet**.** En « concurrence » avec **les méthodes Agiles** et le **Lean Management** (dont on ne parlera pas trop dans ce cours)



**La définition de La gestion de projet** ( = développement par phase/cycle en V)



Phase de Recette : phase de test afin de valider si ce qui a été réaliser est conforme à ce qu’on voulait.

Le but de la méthode de gestion de projet : contrôler le risque -> maitrise des risques (=coûts/délais/périmètres (=fonctionnalités))

En fait le cycle en V c’est ce qu’on apprend à l’école comme méthode de gestion de projet : énoncé de l’exercice = expression des besoins, réalisation de l’exercice = réalisation , correction = recette….;

Le problème avec cette façon de faire c’est qu’elle est très prédictive, on se bloque alors qu’avec le logiciel on a beaucoup plus de souplesse => mieux vos oublié l’idée de projet au sens iso de la chose => méthodes Agiles.

## Manifeste Agile :

<http://agilemanifesto.org/iso/fr/manifesto.html> : Les 4 valeurs

<http://agilemanifesto.org/iso/fr/principles.html> : Les 12 principes

* « Un logiciel opérationnel est la principale mesure d’avancement. » : on fait régulièrement des **réunions d’avancement,** dans lesquels on parle de l’avancement du **projet** (pas du produit hein) pour savoir sur quoi on a avancé, sur quoi on but, dans lesquels on montre un logiciel qui marche (pour les réunions avec les clients) sinon on ne montre pas on parle des incrémentations fonctionnelles.
* À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence. : grosses conséquences sur la façon de travailler l’image du « super-heros » (super hero devloppement) on bosse ensemble.

Software craftmenship : entreprise bordelaise centrée sur agilité hors entreprise ( grosses boites)

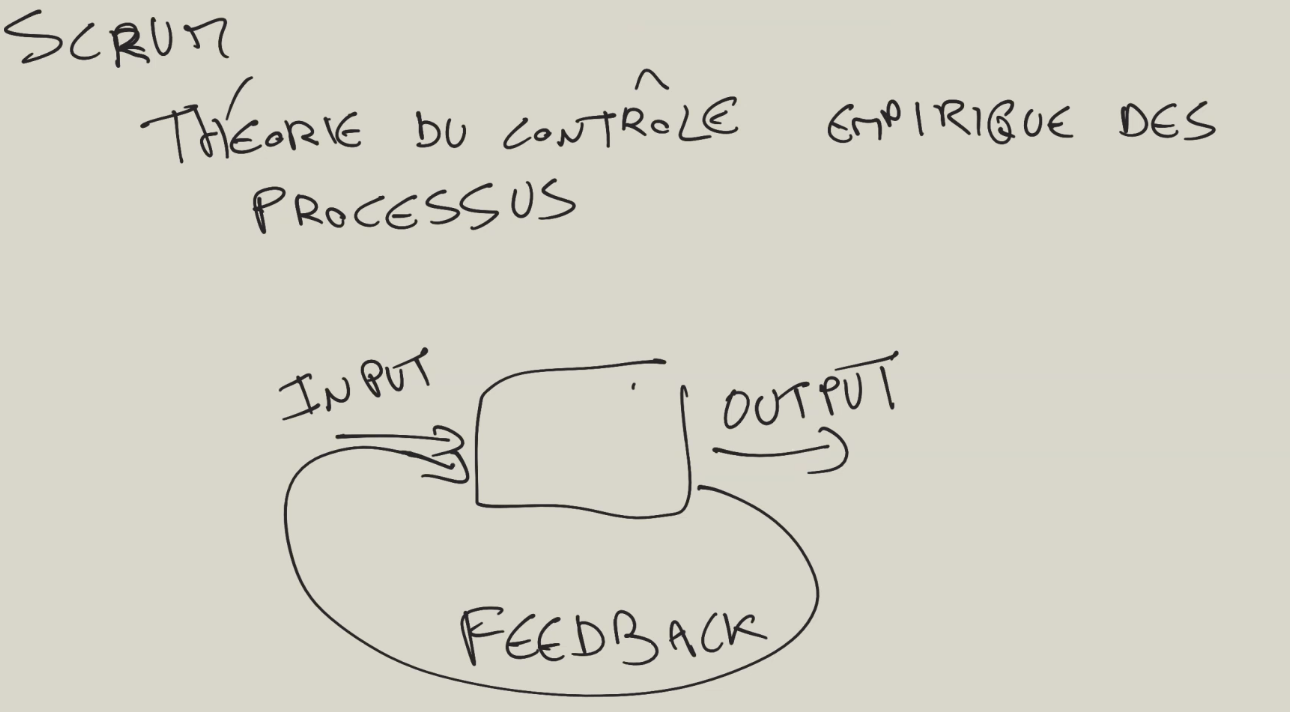
# Méthode(s) Scrum :

Cfs :

* <https://www.scrumguides.org/>
* <https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-French.pdf> : le document de référence ( en FR ) qui explique ce que c’est que Scrum comment le mettre en pratique.
* « Scrum et XP depuis les tranchées » : retour d’experience aux gens qui ont découvert sur le terrain le Scrum et XP (Extreme Programming)

## Le guide Scrum :

« scrum » = mélée (terme du rugby en anglais), terme qui a été trouver bien plus tot chez les japonnais .

* Scrum est fondé sur la **théorie du contrôle empirique de processus**, ou l'empirisme. : mode d’approche basé sur l’expérience par la découverte et l’observation d’une partie des outputs : contrôles en permanence d’une partie de l’output, par du monitoring en permanance
  + 
* 3 piliers :
  + Transparence : on dit si on a fait un gros bug / Dire au client s’il y aura du retard
  + Inspection
  + Adaptation

Scrum est découpé en trois parties :

### Les Rôles :

Un rôle est un rôle pas une fonction ( ca ne dis rien de l’orga de l’entreprise) .

Equipe Scrum [

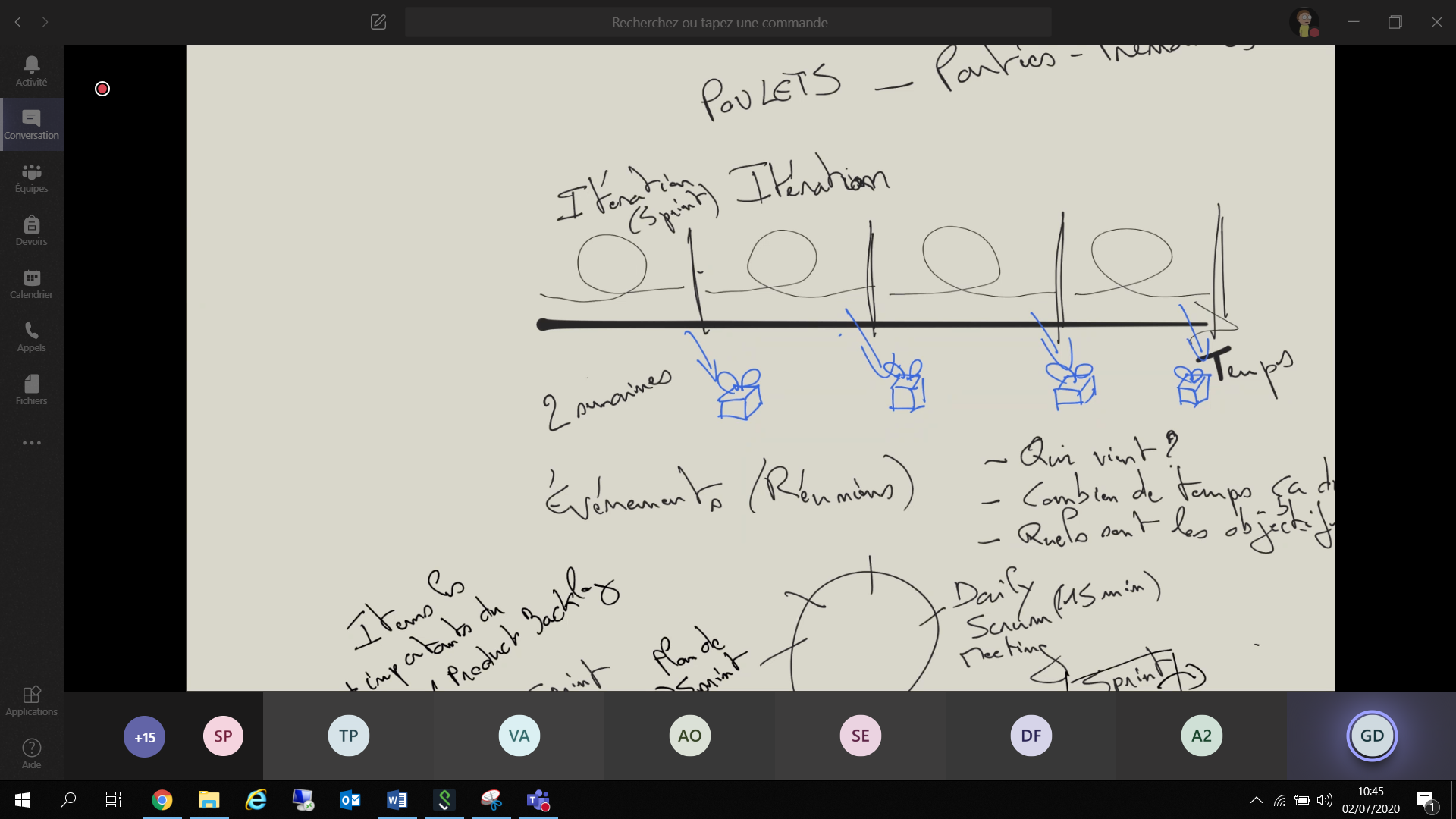
* Le PO (Product Owner = Propriétaire du PRODUIT (pas du « Projet » ca n’existe pas)  ~= Client: Sait ce qu’il va y avoir comme fonctionnalité (fonctionnel) dans le produit, a **le droit de vie et de mort sur chacune d’elles** : définit le produit et de l’importance de ce qu’il y a dedans, réfléchit à la valeur ajoutée de chacune.
  + !!! Proxy Product Owner (PPO) = représente le client, la manager en somme, le chef de projet , ce qui ne respecte pas le principe scrum , méfiance donc ….
  + Il est **unique**
  + un boulot de dingue -> équipe de product owners qui forment le PO avec **un** interlocuteur pour la team dev
* L’Equipe de Développement :
  + Auto organisée : gère tous le « comment » **toute seule**
  + Aucun rôle particulier prédéfini (/!\ les équipes d’experts)
  + Lorsqu’elle est trop grosse on passe généralement en Scrum de scrums (Approche **Nexus**)
  + **Pluridisciplinaire**: un ensemble de personnes qui ont toutes les compétences pour coder les fonctionnalités du produit
* Scrum Master : la pour AIDER les autres à appliquer scrum par la formation, le conseil, les aides logistiques, fait gaffe a ce qu’on ne s’éloigne pas des principes scrum. Protège les devs de l’Organisation en général : <https://www.youtube.com/watch?v=oheekef7oJk>

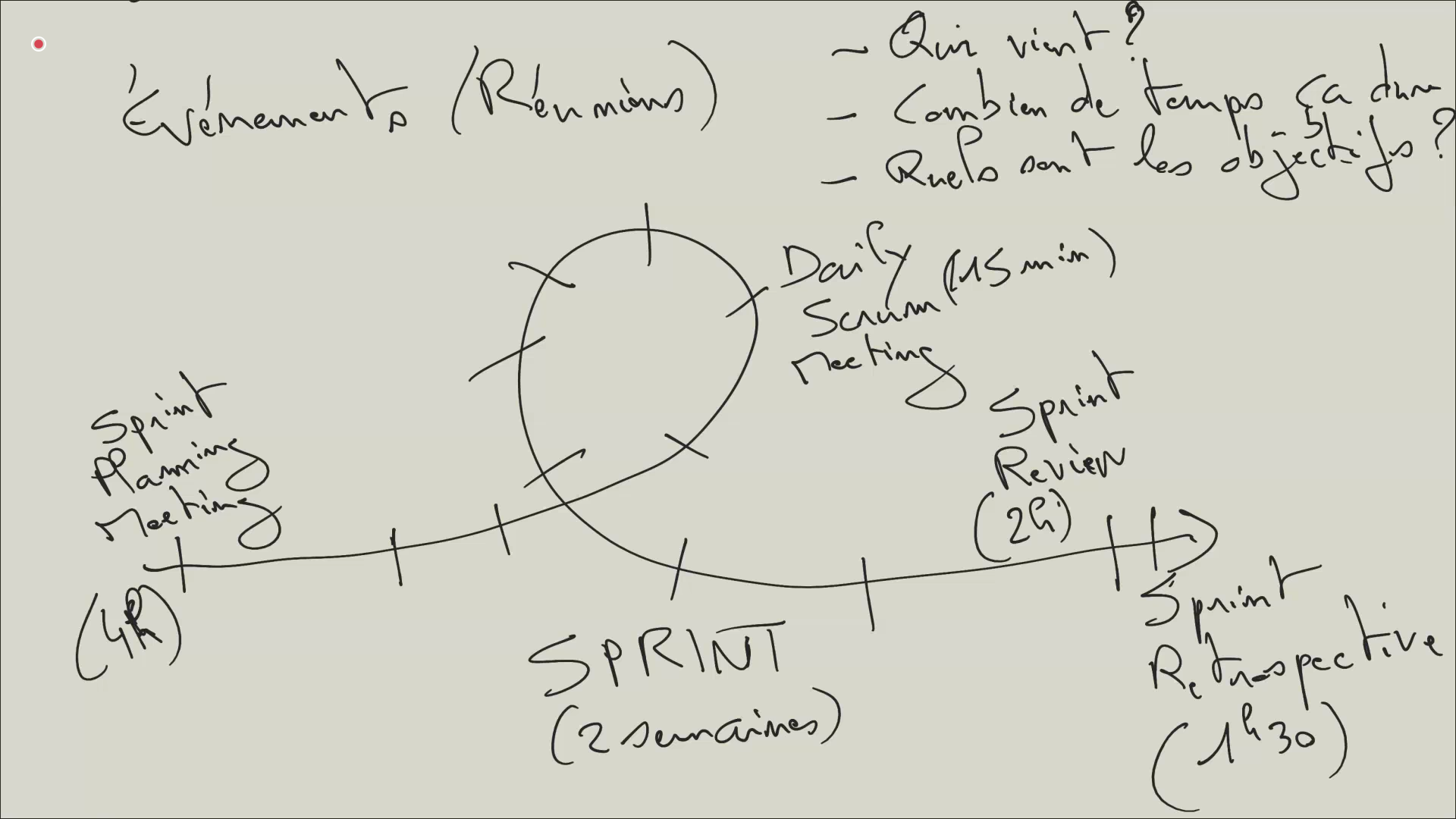
]

* ( Parties prenantes = Stackholders : Ceux qui ne sont ni PO,ni devteam, ni scrum master, tous les autres quoi)

### Les Evènements

Le temps est découper en blocs de temps **fixes** (par exemple 2 semaines) qui se répètent



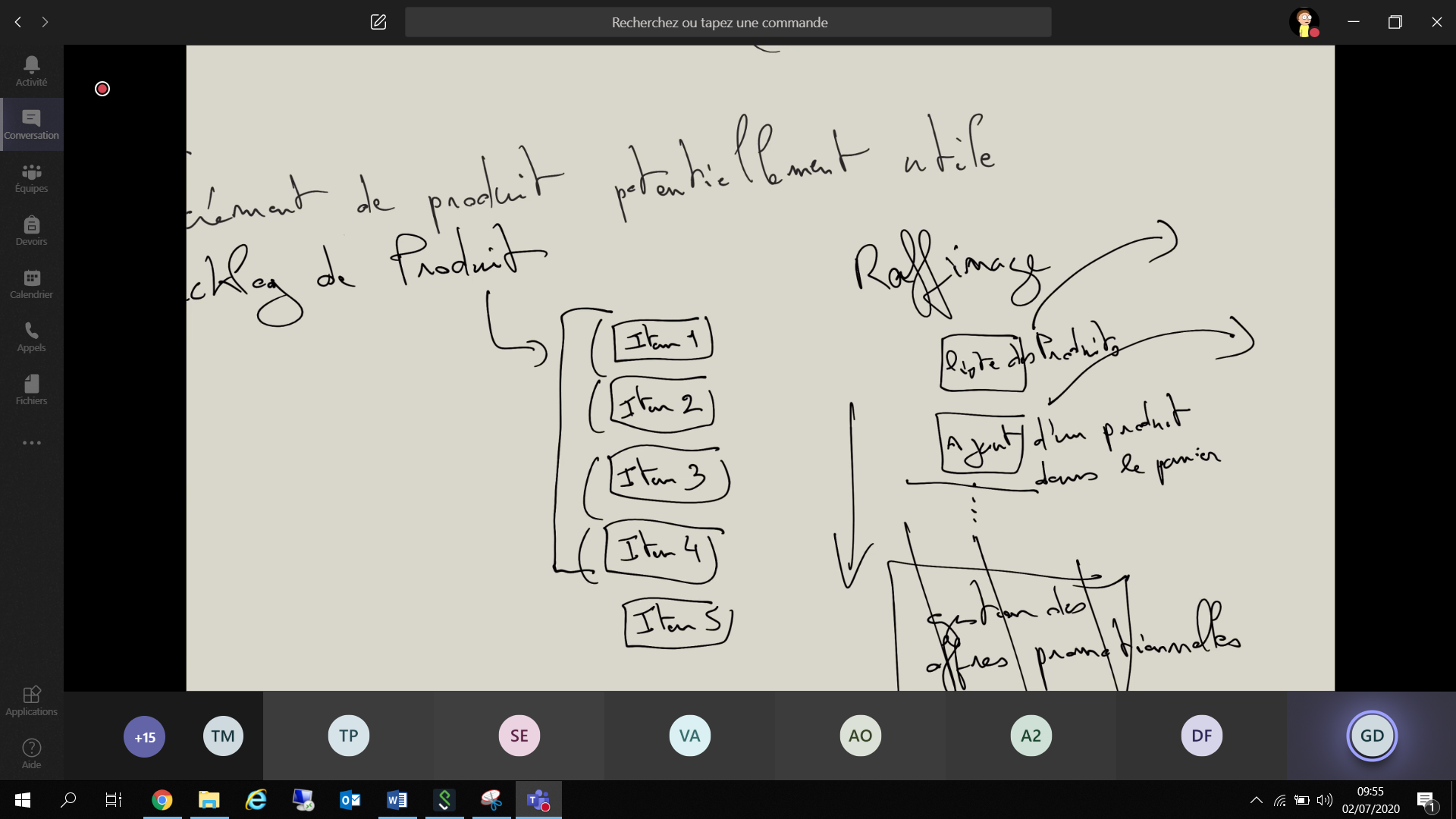
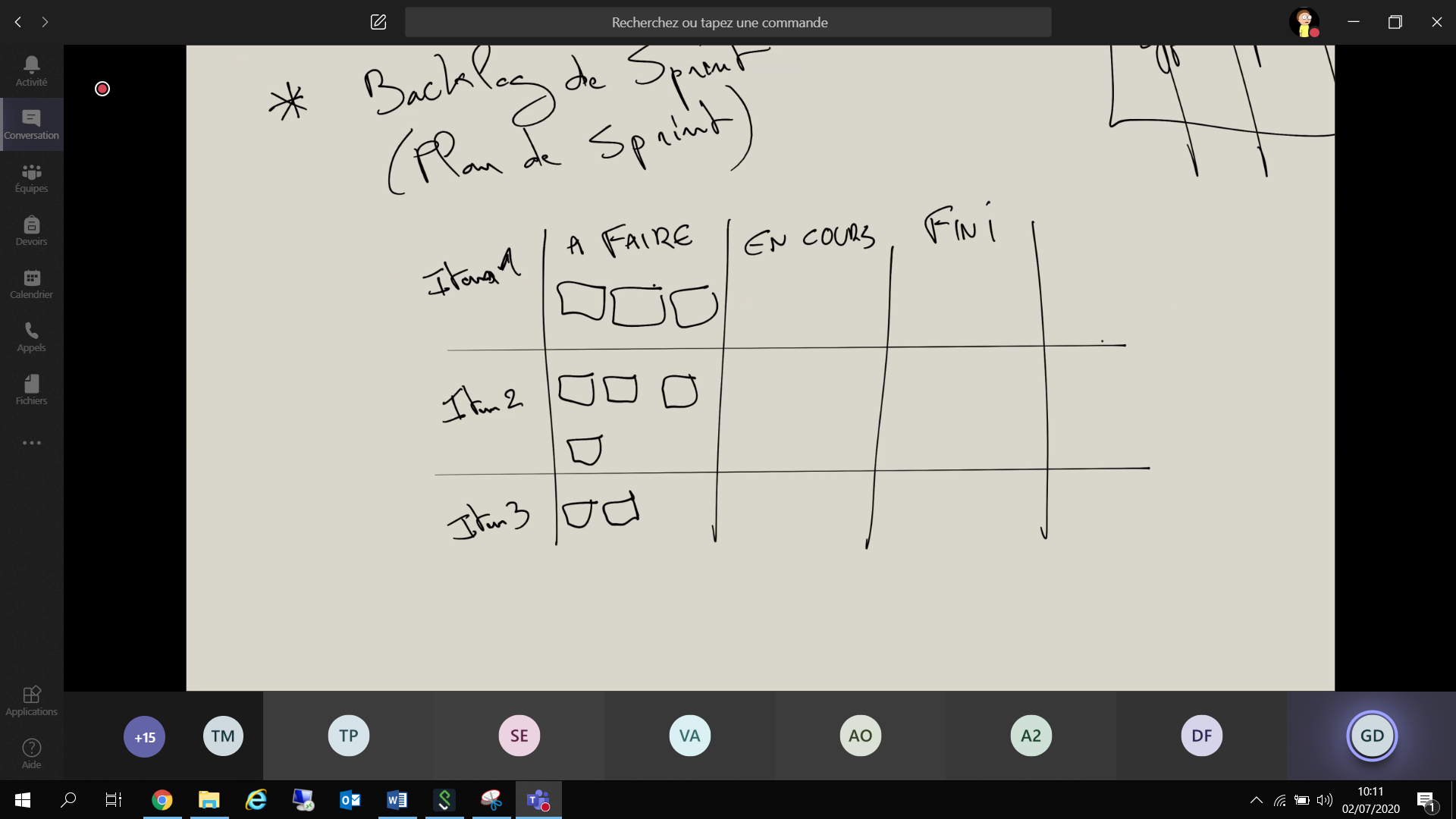


Blocs au cours des quels se répartissent différents évènements : choses qui vont se jalonner pendant le parcours d’un Sprint -> des réunions en somme :

* 3 règles importantes pour ces réunions/évenements :
  + On sait clairement qui vient
  + On sait combien de temps ça dure au max
  + On sait ce qu’on y fait, quels sont les objectifs
* **Sprint Planning meeting**/Réunion de plannification de Sprint (début du Sprint):
  + L’équipe SCRUM au complet (PO,TeamDev,ScrumMaster)
  + Variable selon la durée du sprint (ex : 4h pour sprint de 2 semaines)
  + Que va-t-on faire pendant le Sprint, quels fonctionnalités allons-nous mettre dans le produit et qui sera livrer à la fin du Sprint, la teamDev réfléchit a comment l’organiser voir même commence à découper les tâches
* **Daily Scrum Meeting/Daily meeting :**
  + Toute l’équipe, mais surtout l’équipe de dev (les deux autre sont optionnels)
  + 15 minutes max
  + Permet à l’équipe de se resynchro, voir où elle en est et où elle va. 3 questions a répondre chacuns :
    - Qu’est ce que j’ai fait de puis le dernier daily ?
    - Qu’est ce que je vais faire jusqu’au prochain ?
    - Quels sont les problèmes que je rencontre ?
* **Sprint review / La revue de Sprint** (Fin du Sprint)**:**
  + Toute le monde (Equipe Scrum au complet + les stackholders)
  + Pour 2semaines : 2h
  + Faire la review du **Produit** (= une démo) pour constater l’évolution du produit et faire des remontées
* **Sprint rétrospective/La rétrospective du Sprint :**
  + Equipe SCRUM
  + 1h30 pour des sprints de 2 semaines
  + Comment on a bossé, est ce qu’il y a des chose à garder/changer etc…

### Les Artefacts

Ce qui est produit par l’equipe Scrum :

* Le **produit** lui-même
* L’incrément de produit potentiellement utile
  + C’est pour ça qu’on commence généralement la mise en place/developpement de l’environnement de production dès le début du sprint (Fail Fast) pour éviter la galère à la fin (du sprint) au moment de la livraison
* **Le Backlog de produit** : outil principale de travail du PO
  + Vit et évolue au fil du développement et de la vie du produit
  + Contient des Items : liste de fonctionnalités voulues dans le produit
  + Organisés/priorisé par le PO
  + Liste des bugs
  + Technique du raffinage : prendre la décision le plus tard possible pour la prendre au bon moment : consiste à dire : que des fonctionnalités les plus importantes génère des items mises en haut du backlog (ex : liste articles, ajout article au panier) les moins importantes (ex : gestion des offre promotionnelles) sous regrouper sur une « fiche » qui remontera dans le backlog au fur et a mesure de l’avancement du produit et qui une fois vers le haut sera découpée en plusieurs items : 
  + En gros c’est la flamme qui tient le produit vivant
* **Backlog de Sprint (**Plan de Sprint**)**:
  + Contient les fonctionnalités/items a développé/livrer a la fin du Sprint
  + C’est l’output du Spring planning meeting
  + 
* **(Objectifs de Spring : )**
  + Peuvent être fonctionnels, qualitatifs …
  + Ce concentre surtout sur l’efficience du « projet »
* **Définition de « fini » / Définition of « done » :**
  + Check-list qui définit ce qu’il faut avoir fait sur un item pour le considérer comme « Terminer »

## Petits mots sur l’Extreme Programming (XP)

Réfléxion de Kent Beck sur son métier avec tous ce que ça implique

Met l’accent sur : tous les domaines (planification, organisation, technique ….) contrairement au Scrum qui met l’accent surtout sur l’organisation laissant de coté le reste a l’intepretation et aux préfèrence des applicants

Pourquoi ce nom : Dire on constate qu’il y a des pratiques plutôt utiles pour la dev team donc « **soyons extreme tout le temps** » et utilisons toutes ces pratiques (retour utilisateurs, livraisons, etc…) **Tout le temps**.

Exemple de pratique :

* Intégration continu
* Déploiement continue
* Peer programming
* User stories

## Petits mots rapide sur DevOps

Ni un outilage ni un nouveau métier/ personne multi-fonction

#DevOps : tous la partie d’Agile qui s’intéresse à l’administration système du produit au sein d’un projet en mode Agile. Toute l’équipe s’intéresse et maitrise aussi la partie admin system. Ce qui a conduit entre autre a l’automatisation d’infrastructures (Docker et Ansible par exemple), l’utilsation de gestionnaires de sources (ex : Git, Subversion), « l’infrastructure as code ».

# Estimation – Planification – Priorisation

Expression des besoins, c’est la base.

Dans Scrum elle est materialisée par le Backlog Produit et ses items

Dans XP, c’est par les **User Story** (US = histoire qui met en scène un utilisateur et ses interactions avec le système)

Du coup dans le Backlog Produits les items sont au départ souvent fais d’user story

Une User story est exprimée de cette manière le plus souvent :

1. « En tant que ……… »
2. « Je fais ……… »
3. « Afin de …….. »
4. Saisi dans un outil de Backlog

Pour que les programmeurs comprennent le sens des items qu’ils développent.

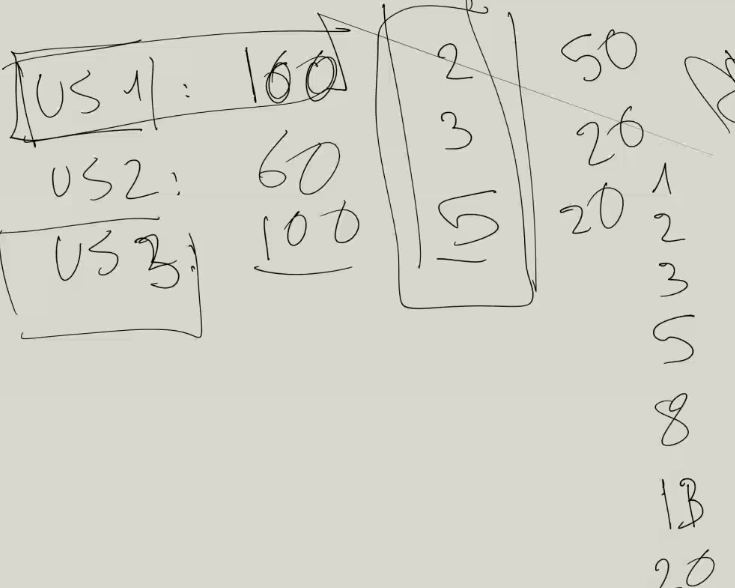
Bonne manière de faire des user story : **INVEST** :

* **Indépendant** : Si on fait ou pas cette US ça ne doit pas avoir d’incidence sur le reste du système | les US doivent être indépendantes les unes des autres (indépendance relative tout de même ) <= le plus discutable
* **Négociable** : entre celui qui la raconte et celui qui l’écoute / entre le PO et le développeur. De facon a la mettre la plus rapidement en place
* **Valuable**: il faut qu’elle est une valeur métier (Indicateur)
* **Estimable**: Estimable en terme de difficulté/complexité technique (Indicateur)
* **Small**: Elle doit être courte et concise, raffinée, suffisamment petit (**1 ou 2 jours max de dev** c’est donc les devs qui délimitent ca). Sinon c’est se qu’on appel une « Epopée ». Si trop grande, comme toute grosse charge travail, il faut découper.
* **Testable :** tests d’intégration, test d’acceptance etc… pour savoir que cette US est prête a être livré

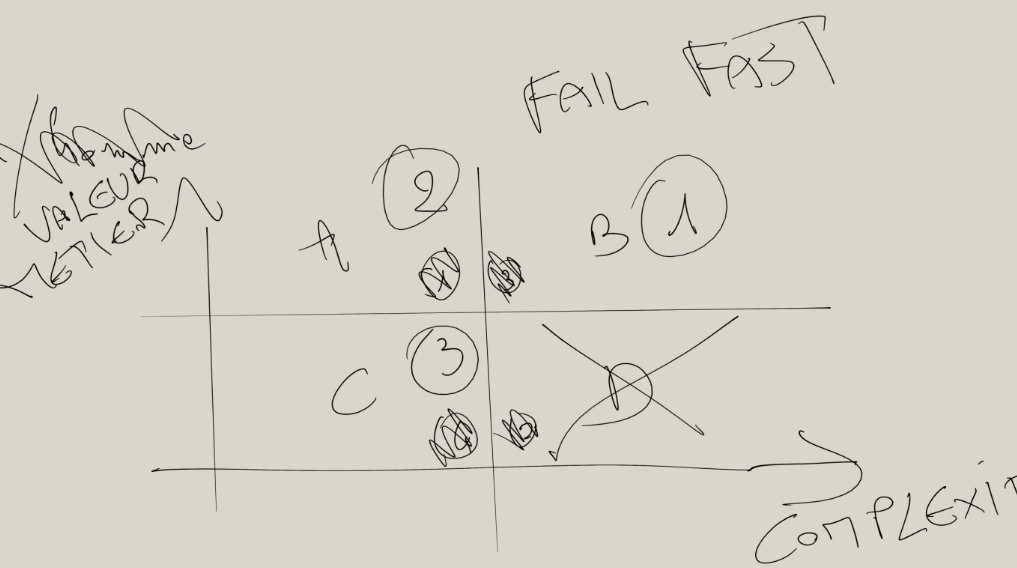
Cf : ./exemples US/\*

Valuable : Mettre des indicateurs numériques qui permettent de comparés les items desquels découlent les US. Selon trois critères :

* Valeur métier (fait par le PO)
* Priorité
* Complexité (-> **Estimable**) : La complexité est déterminer soit en jours/homme (le nbr de jour nécessaire a dev cet item si il était dev par une seule personne) ou avec des indicateurs numérique qui suit la suite de Fibonaccie toujours affecter par la Team Dev



Ensuite l’ordre d’execution/la priorité, de ces fonctionnalités ce definit un peu comme ca ( Methode McKone)



Ca donne ca avec des Sprint ( en partant du premier Sprint du produit) et on peut calculer un troisièe indicateur : la vitesse moyenne de l’équipe = la **Vélocité.** Ce qui permet au PO de faire de la planification a relativement long terme sur l’avancement du produit