

# Projet GL

## Gestion de Projet

Ensimag 1A Alternants  
Philippe Bodiglio

# Les attentes

- Mettre en place une organisation d'équipe efficace qui intègre tous les membres de l'équipe
- Mettre en place un planning prévisionnel et suivre l'avancement du projet (mesure des écarts et mise en place d'actions correctrices si nécessaire)
- Communiquer sur votre projet aussi bien à des spécialistes qu'à des non-spécialistes, et à la fois sur le contenu technique et la gestion de projet.
- Faire un bilan de votre projet : en tirer des leçons à la fois de vos erreurs et de vos réussites !

# UN TOUR D'HORIZON DE LA METHODE SCRUM

# Introduction

Vidéo : « SCRUM pour les nuls »

*(source Moullouze Mohamed – You Tube)*

<https://www.youtube.com/watch?v=kZTLIWkxFN4>

Vidéo : « Intro to Scrum in Under 10 Minutes »

*(source Axosoft – You Tube)*

<https://www.youtube.com/watch?reload=9&v=XU0IIRltyFM&feature=youtu.be>

**D'accord, mais comment est-ce  
que l'on fait ça sur le projet GL ?**

**Dans les promotions  
précédentes, certains ont bien  
réussi**

# Comment ont-ils réussi ? (les promotions précédentes)

## C'est quoi réussir ?

Faire le bon produit au bon niveau de qualité,  
et bien faire tout ce que l'on nous a demandé.

**OK... mais pas que**

# Réussir c'est aussi...

## Grandir individuellement ET en équipe

- Apprendre à faire,
- apprendre à mieux travailler ensemble, s'entraider
- savoir être lucide, regarder la réalité,
- découvrir comment s'améliorer, et le faire,

**Savoir mettre en place un cadre et une qualité de relations qui permettront d'évoluer dans toutes les circonstances**



# Ceux qui ont le mieux réussi :

- Ont su découper le fonctionnel en user stories,
- Ont été bon dans le suivi quotidien, les stand-ups et l'attention aux problèmes des coéquipiers
- Se sont bien conformés à leur définition de terminé
- Ont testé, testé et testé
- Ont fait un burndown chart (même s'ils n'y viennent qu'au dernier sprint)

# LA GESTION DU CONTENU

## DÉCOUPER LE TRAVAIL

Zoom sur les User stories



# Pour bien faire ça, il faut éviter de manger le gâteau par couche



Analyse

Architecture

Code

Données

# Mangez plutôt des petites parts



Analyse

Architecture

Code

Données

# Pratique : les user stories

user story = user + story

(histoire utilisateur = histoire + utilisateur)

Il y a deux parties dans une user story

- Un utilisateur
- Une histoire

Représente les utilisateurs de votre produit

Raconte ce pour quoi ils s'en servent



# Exemple 1

**En tant qu'utilisateur du compilateur deca**  
**Je souhaite** pouvoir compiler un programme avec une fonction vide

**De façon à** pouvoir valider que mon compilateur respecte la syntaxe Deca sur les fonctions vides

## Conditions d'acceptation

Si j'écris une fonction avec une syntaxe correcte et qui ne contient aucune ligne de programme, le compilateur l'accepte.  
Si j'écris une fonction avec une syntaxe incorrecte, le compilateur sort une erreur



## Exemple 2

**En tant que** qu'utilisateur du compilateur  
**Je souhaite** pouvoir tester l'inégalité entre deux valeurs numériques comme condition d'un « if »  
**De façon à** pouvoir faire deux calculs différents selon les valeurs

### Conditions d'acceptation

Une condition de « if » avec une inégalité est acceptée entre entiers et flottants(deux entiers, deux flottants, ou un entier et un flottant).

Une condition avec inégalité n'est pas acceptée entre deux types objets.

Une condition peut comparer des variables, des champs ou des paramètres aussi bien avec des constantes, qu'avec d'autres variables etc.

Si la condition est satisfaite, je dois exécuter la branche if (et pas la branche else)

Si la condition est fausse, je dois exécuter la branche else (et pas la branche if).



# Évaluer l'effort de réalisation des story points en utilisant le « planning poker » »





# Évaluation

## Prétexte à une discussion en équipe

- Quel travail à faire sur cette part ?
- Qu'est ce que l'on connaît ?
- Qu'est-ce qui est facile ou difficile
- Est-ce qu'on a les données de test ?
- ...etc.

**1 jeu = 8 cartes**

1,2,3,5,8,13,21 et « infini »  
(suite de Fibonacci)



# Évaluation

**Préférer des stories de petite taille**



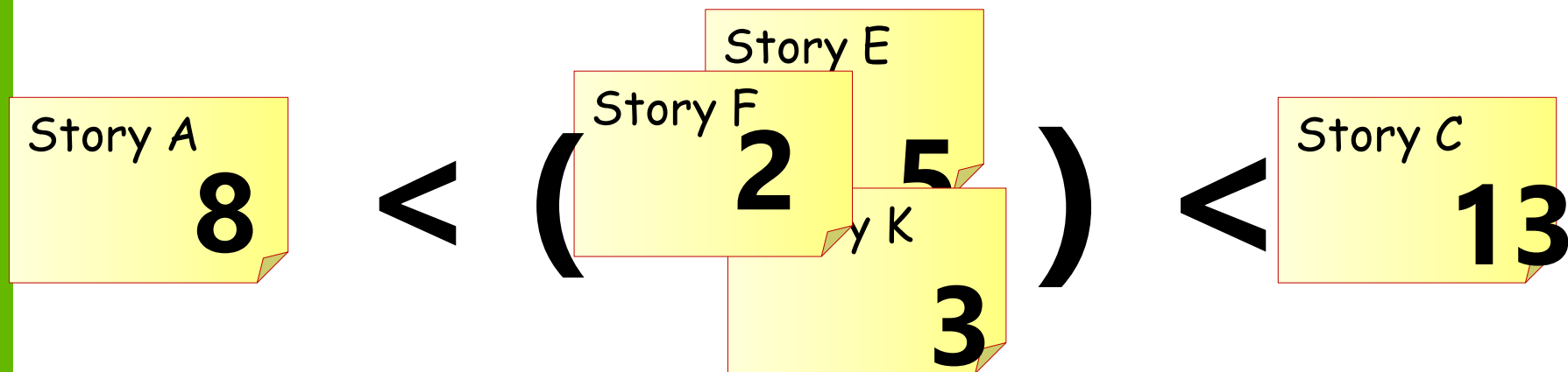
**La taille c'est la quantité d'effort demandé**

- 1 : très facile à faire
- 20 : énormément d'efforts à produire (très risqué dans un sprint)
- > 20 : il faut découper on ne va pas y arriver

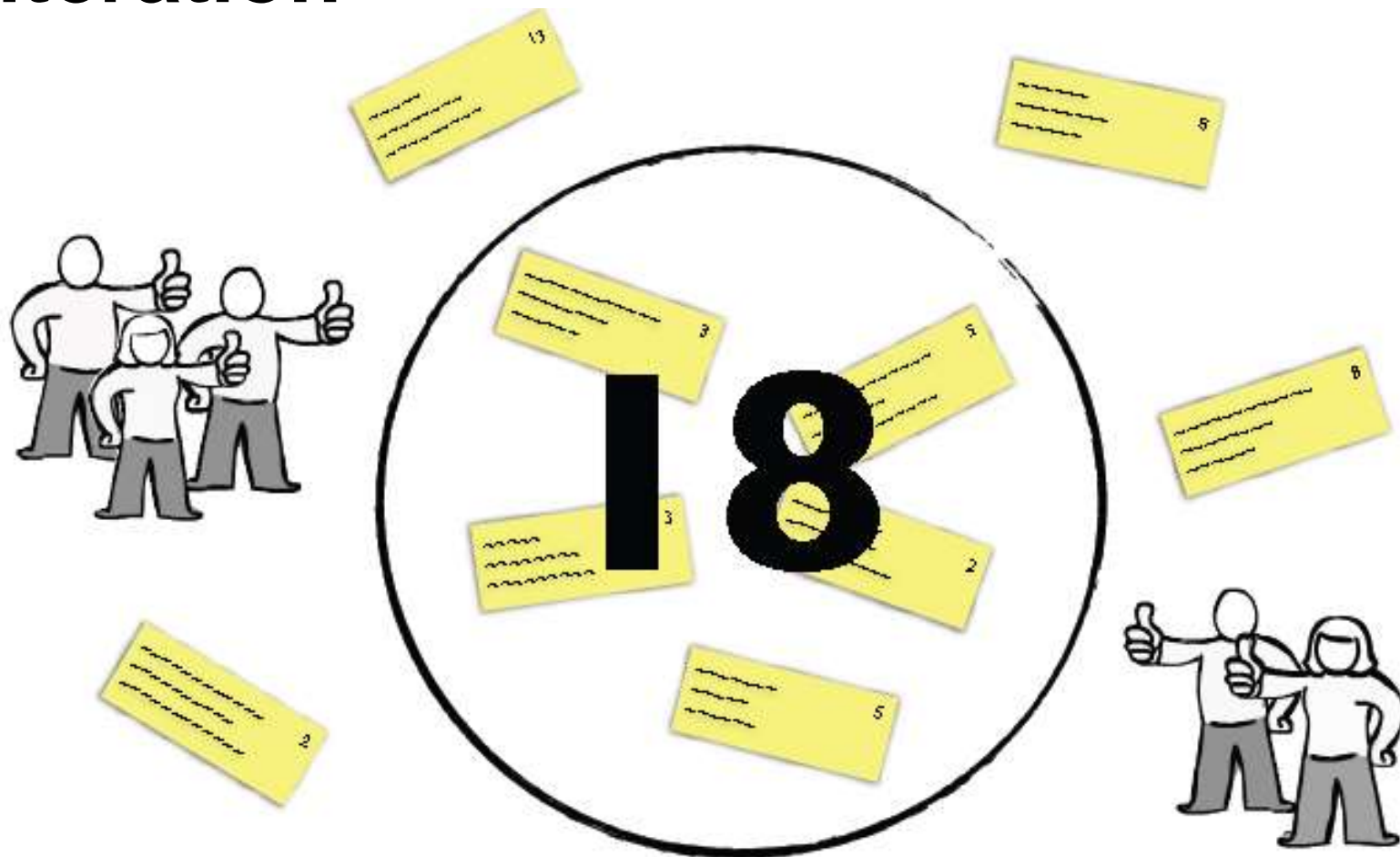
**Dans une itération les équipes produisent en moyenne de 5 à 8 stories**

# Évaluation

- Ces chiffres ne vous donnent pas le temps nécessaire à la réalisation
- Mais si vous faites bien le job, un 5 sera toujours ~ à un autre 5, un 8 à un autre 8...
- Les règles arithmétiques de base s'appliquent



# C'est l'expérience qui vous dira combien vous pouvez produire par itération



# Attention !

## ceci n'est pas une user story

**En tant que** développeur

**Je souhaite** mettre en œuvre un analyseur syntaxique pour Deca

**De façon** à pouvoir construire l'arbre abstrait primitif du programme ( et savoir faire l'étape A : Analyse lexicale et syntaxique ).

**Peut mieux faire**

# Au démarrage du sprint

Découper en tâche les stories qui entrent dans le sprint

**Story 1 - 5 points** - Affectation variable entière nommée sur 1 char

**NB: sur les tâches on peut mettre les heures estimées**

Analyser le support de projet et en ressortir les quelques éléments pertinents (restreindre au nécessaire)

4h

Récupérer les bibliothèques

8h

Écrire le script de test

2h

Coder la fonction xxx

8h

Tester et valider

8h

Documenter la procédure

4h

# Travailler ensemble


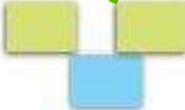



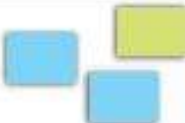

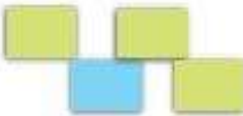


Les outils pour le suivi de la réalisation

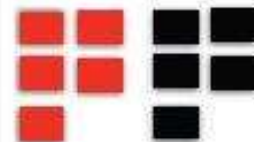
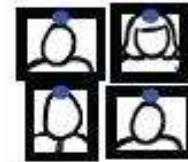


# Un tableau des tâches pour visualiser le flux

Stories en  
pts

Taches en h

Stories	A faire	En cours	Fini
			
			
			
			
			



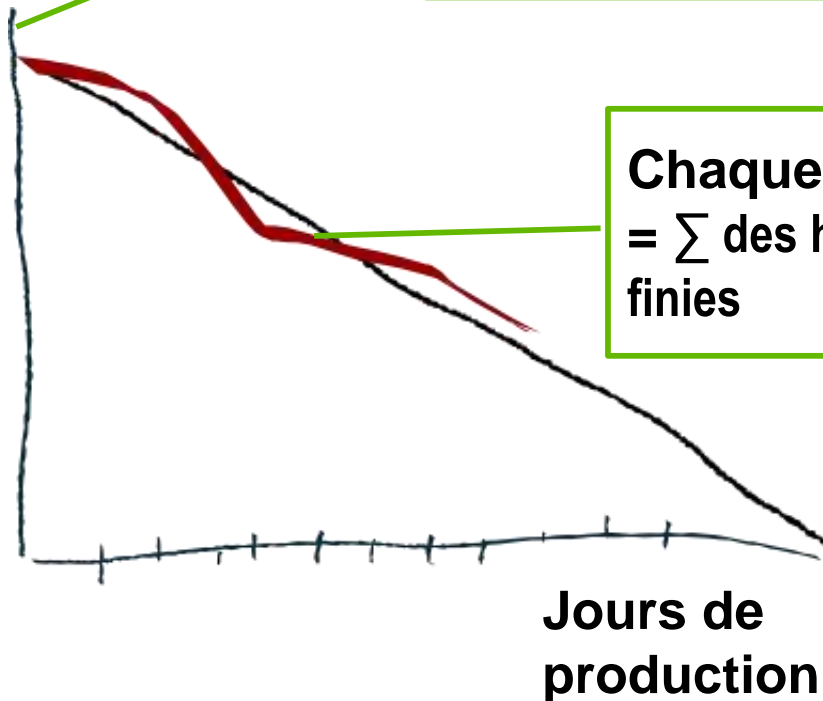


# Suivre le travail avec le « burndown »

Somme des heures des tâches non finies

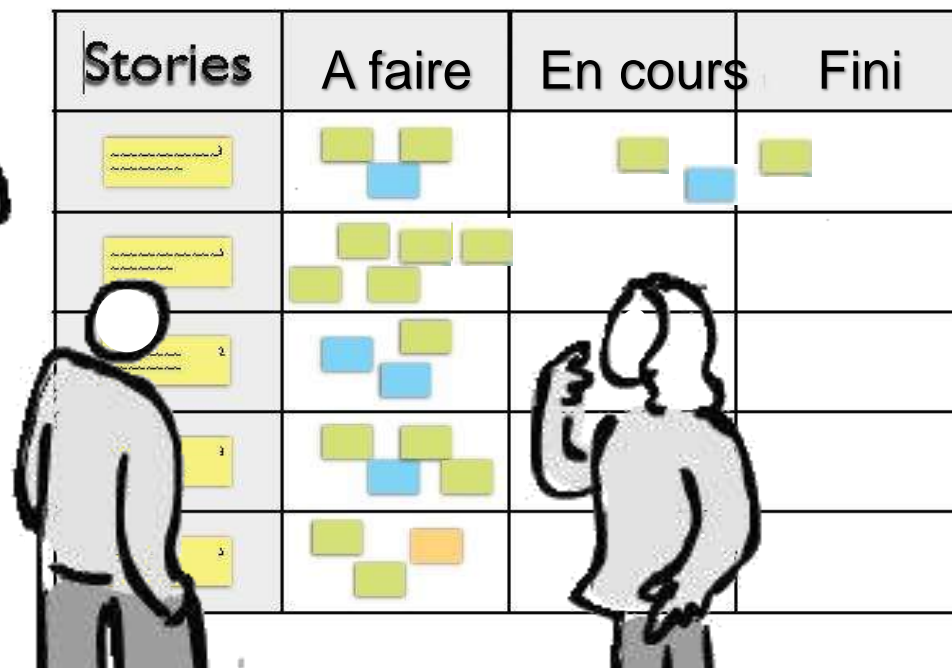
Au début tout est à faire =  $\sum$  des heures de toutes les tâches

Chaque jour relevé du réel =  $\sum$  des heures des tâches non finies



# « stand-up » journalier

**20mn maximum**  
**Pas de discussion**



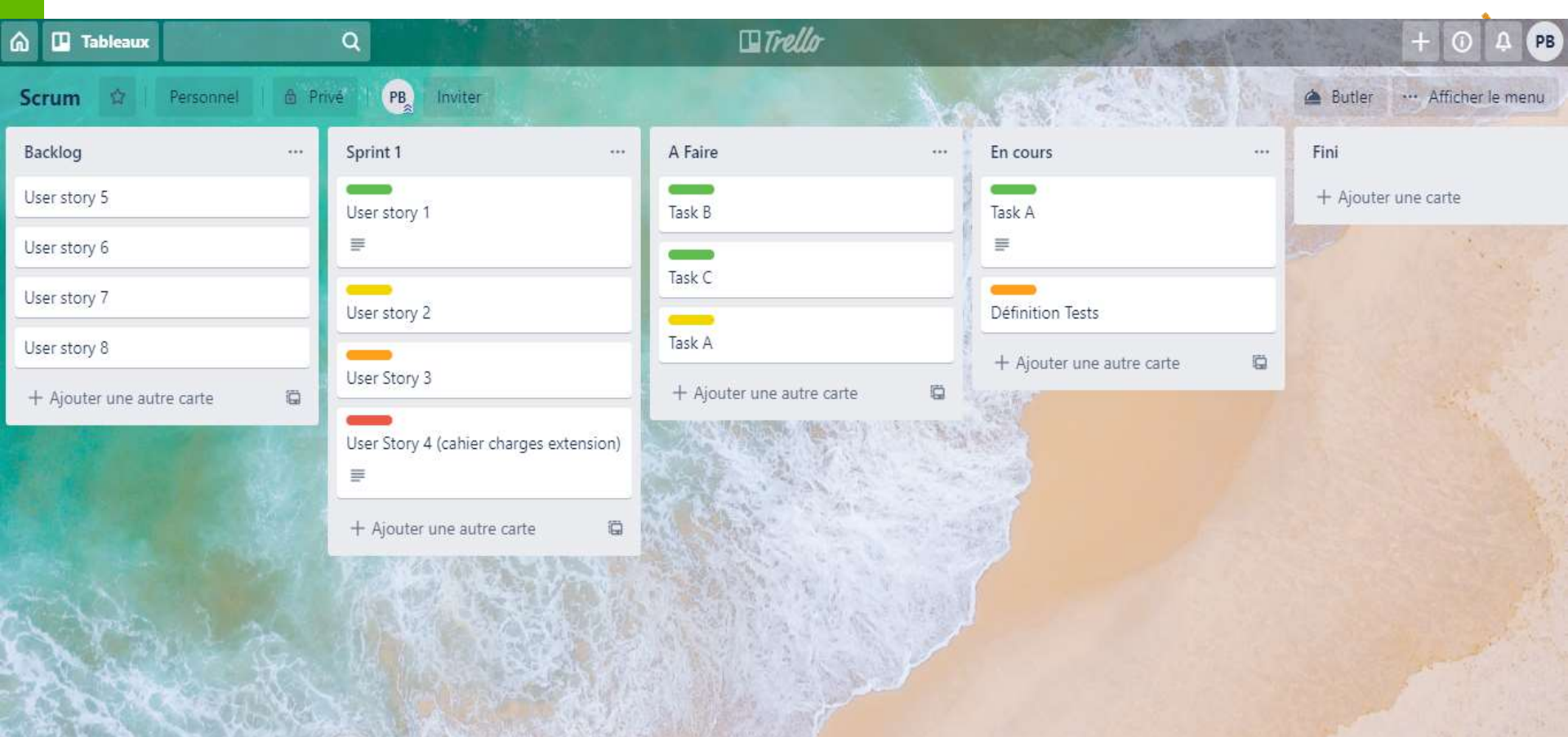
**Chacun à tour de rôle énonce trois phrases**

- Hier j'ai fait...
- Aujourd'hui je vais faire...
- J'ai / je n'ai pas / de problème

# Les outils possibles : Trello

Tableau Trello :

- créer une liste pour le **Backlog** (projet complet) et chacun des **Sprints**
  - dont les cartes sont les **User stories**
- et,
- créer trois listes : **A faire, En cours , Fini**
  - dont les cartes sont les **Tâches**
  - utiliser le label de la carte tâche pour la lier à une user story



## Limites :

- Pas d'estimation des tâches
- Pas de gestion du burndown chart

# Les outils possibles : Gitlab

- Utiliser les objets Board et Milestone pour le **Backlog** et les **Sprints**
- Utiliser les labels A faire, En cours , Fini
- Utiliser les issues pour les **Tâches** et **User stories**

*(voir slides Scrum-Gitlab - Olivier Alphand - dans Chamillo)*

## Avantages / Limites :

- Intégration Git
- Possibilité de burndown mais sans notion de « reste à faire »

# Les outils possibles

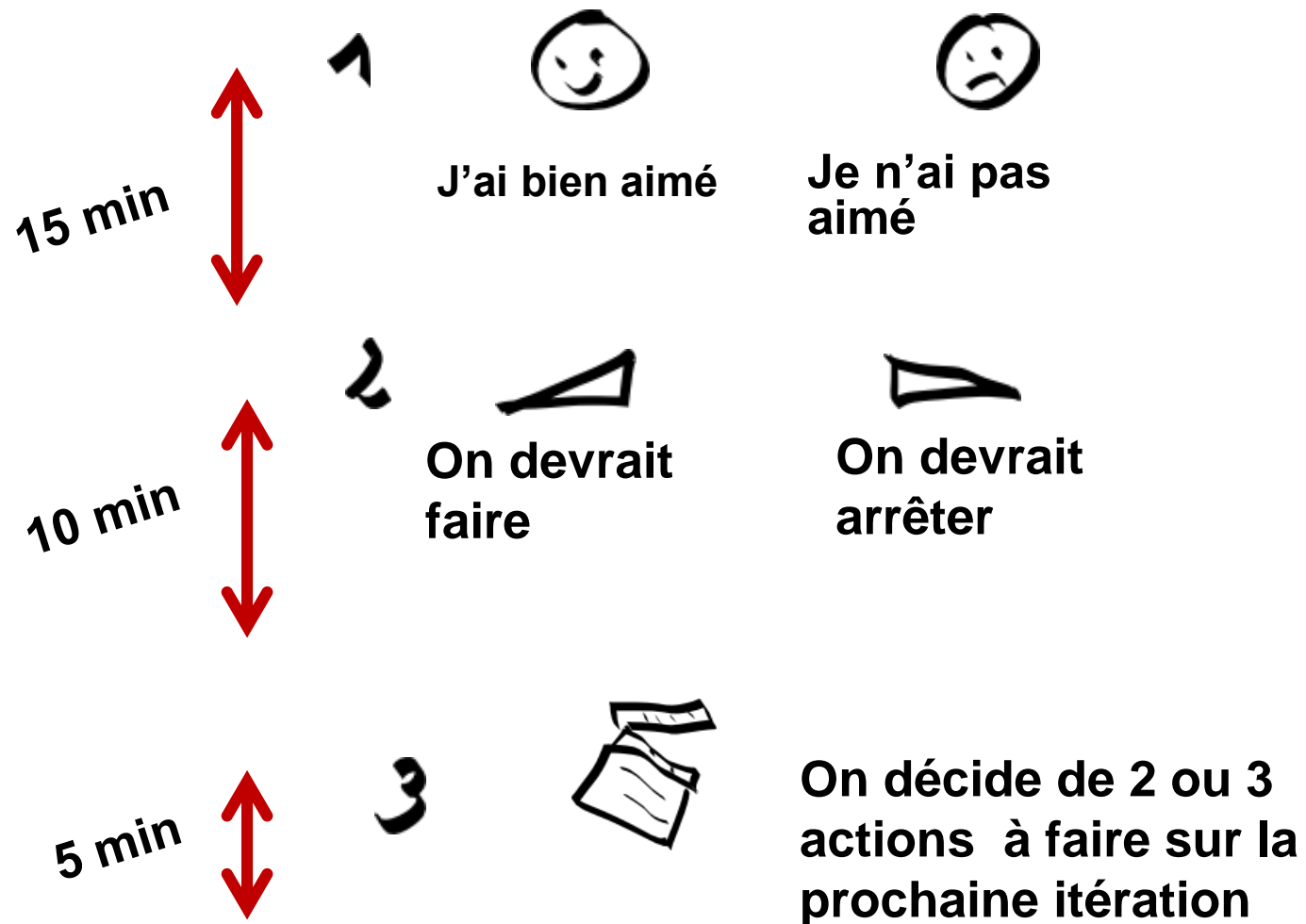
Nous sommes preneurs de toute proposition de bonnes pratiques pour les générations futures !

*Avec Trello, Gitlab ou d'autres outils gratuits*

**A la fin de l'itération, faire une démo et valider ce qui est fini**



# Ensuite, faire la rétrospective tous ensemble





**Travaillez bien, et  
surtout :**

**Amusez-  
vous !**

