

# INFOF-307 - GÉNIE LOGICIEL ET GESTION DE PROJETS

- EXTREME PROGRAMMING -  
- GESTION DE PROJET -

Pluquet Frédéric

Assistants: A. Cnudde, A. Laval, Y. Molinghen, A. Reynouard

2021-2022

ULB

# SOMMAIRE

- Introduction
- **Gestion de projet**
- Gestion de l'équipe
- Gestion de la qualité du code
- Autres considérations
- Conclusions



# GESTION DE PROJET

- Principes de la démarche
  - Rechercher le rythme optimal
  - (Re)définir régulièrement le projet
  - Maintenir l'équilibre entre développeurs et client
  - Définir les spécifications tout au long du projet

# DÉMARCHE

## RECHERCHER LE RYTHME OPTIMAL

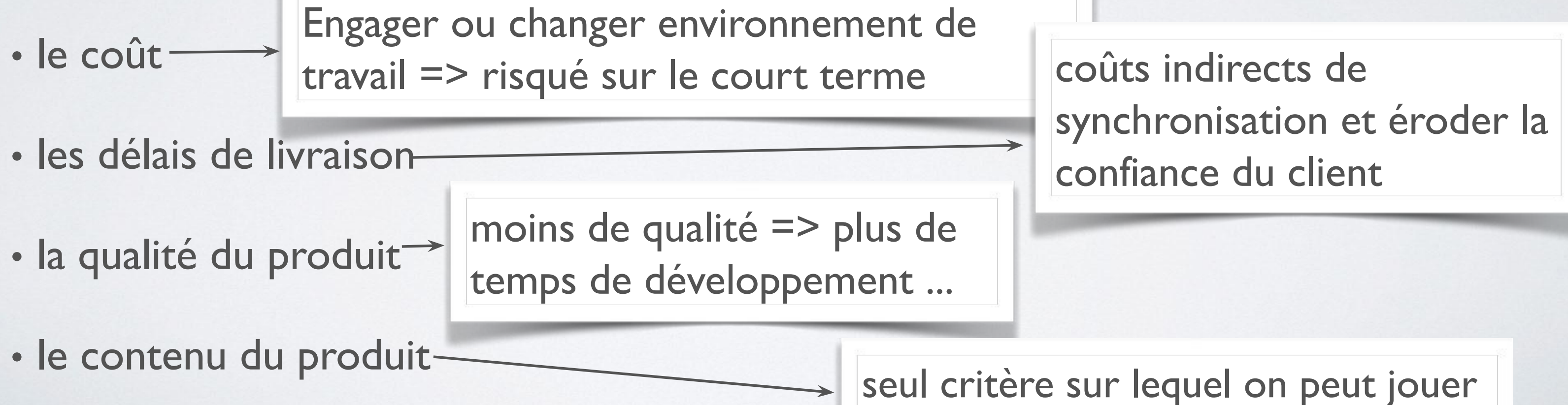
- Si on ne prend pas le temps:
  - Mauvaise qualité de code
    - => code moins maintenable
      - => plus de temps pour chaque changement
    - + de régression
      - => Moins bonne qualité de code



# DÉMARCHE

## RECHERCHER LE RYTHME OPTIMAL

- Contrainte entre rythme optimal de travail et contraintes de l'entreprise
- Un projet est dimensionné par 4 variables:



# DÉMARCHE (RE)DÉFINIR RÉGULIÈREMENT LE PROJET

Méthodologie **waterfall**:

«**Visez... Feu ! Raté...**»

Méthodologie **XP**:

«**Feu ! Visez... visez... visez...**»



# DÉMARCHE

## (RE)DÉFINIR RÉGULIÈREMENT LE PROJET

- I. Commence par une phase initiale
  - Exploration du problème
  - Recenser les besoins du client
  - Granularité assez large
    - Choix techniques initiaux
    - Estimer les coûts de manière fiable

# DÉMARCHE (RE)DÉFINIR RÉGULIÈREMENT LE PROJET

- 2. Plan de livraisons
  - Basé sur les estimations de la phase initiale
  - Le client doit définir finement les priorités des fonctionnalités
    - Les plus importantes en premier
    - Disponibles rapidement



# DÉMARCHE

## (RE)DÉFINIR RÉGULIÈREMENT LE PROJET

- 2. Plan de livraisons
  - *XP: Ce plan pourra changer en cours de projet (ajout d'une tâche, suppression, repousser, ...)*
    - Le client s'appuie sur l'expérience acquise, l'évolution de l'application
    - L'équipe peut estimer de mieux en mieux les coûts de la prochaine itération
  - => on joue bien sur la variable *contenu* du projet

# DÉMARCHE

## ÉQUILIBRE ENTRE DÉVELOPPEURS ET CLIENT

- Tout ça fonctionne si le client et les développeurs jouent le jeu
- Si le client a trop de pouvoir
  - il peut demander l'impossible aux développeurs
- Si les développeurs ont trop de pouvoir
  - ils peuvent s'enliser dans un projet générique qui n'apportera rien au client



# DÉMARCHE

## ÉQUILIBRE ENTRE DÉVELOPPEURS ET CLIENT

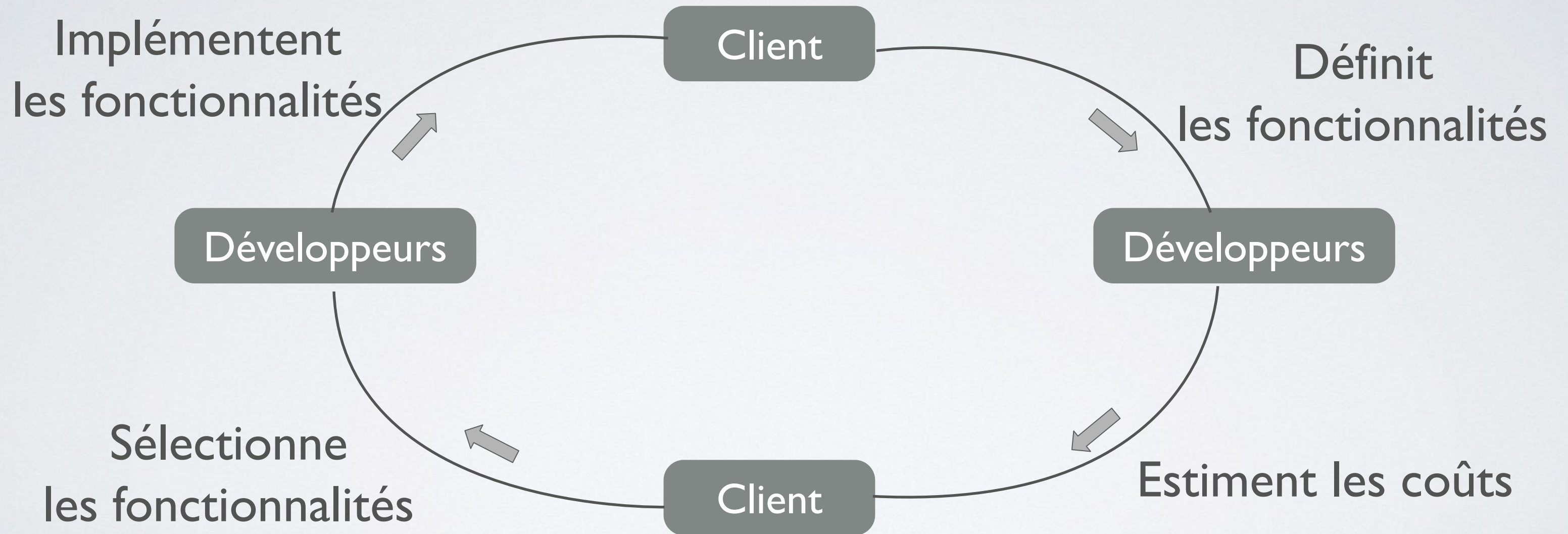
- Pour garder l'équilibre, XP impose une séparation stricte des responsabilités

Le <b>client</b>	Les <b>développeurs</b>
définit les fonctionnalités décide de l'ordre d'implémentation	fournissent les estimations de coûts se chargent de la réalisation des fonctionnalités

- (voir chartes dans la chapitre suivant)

# DÉMARCHE

## ÉQUILIBRE ENTRE DÉVELOPPEURS ET CLIENT





# DÉMARCHE

## DÉFINIR LES SPÉCIFICATIONS

- Les activités d'analyse et de spécification sont échelonnées tout au long du projet
  - Activités quotidiennes des développeurs
  - Comme programmation et tests
- Mais si pas de conception: comment définir l'organisation de l'application ?
  - Via la conception simple, le remaniement et les tests unitaires
  - L'architecture émerge au fur et à mesure des besoins

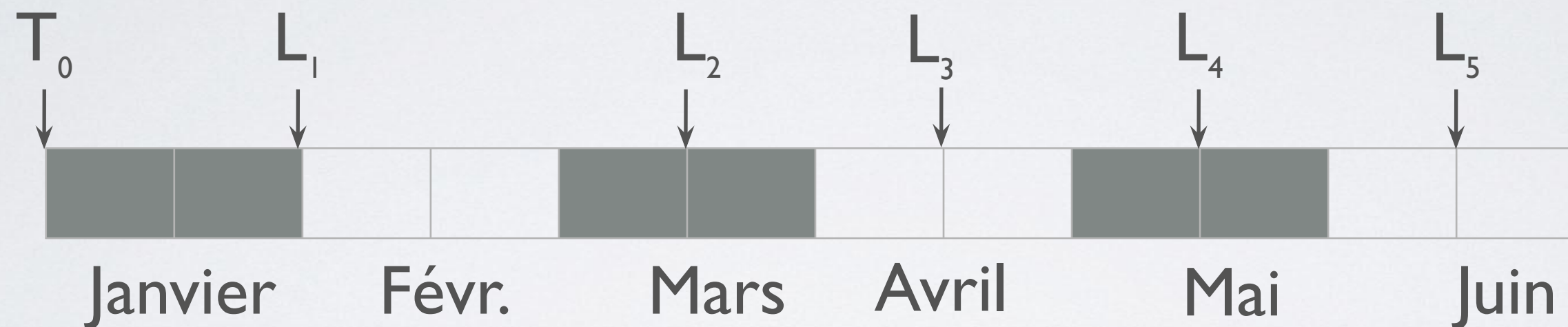
# PRATIQUE XP DE GESTION DE PROJET

- Livraisons fréquentes
- Planification itérative
- Client sur site
- Rythme durable



# LIVRAISONS FRÉQUENTES

- Exemple



Deux idées clés:

- *Première livraison très rapide*: éviter les malentendus et donner de la consistance au projet
- *Livraisons suivantes aussi rapprochées que possible*: pilotage précis et preuves fréquentes de l'avancement

# LIVRAISONS FRÉQUENTES

- Poussées à l'extrême
  - livraisons quotidiennes:
    - grâce à l'intégration continue et tests automatiques
    - exemple: application web
- Limitation de la fréquence si lourdeur du processus de livraison
  - Si validation externe, ou logiciel embarqué dans un téléphone par ex.



# LIVRAISONS FRÉQUENTES

- Les livraisons fréquentes permettent un meilleur *feedback* du client
  - Peut mieux cerner les besoins pour les livraisons suivantes
- Feedback pour l'équipe
  - Sentiment régulier de « travail fini »
    - Entretient la motivation et diminue la pression
  - Le logiciel est confronté à un environnement réel (temps de réponse sur le web plus long qu'en local)



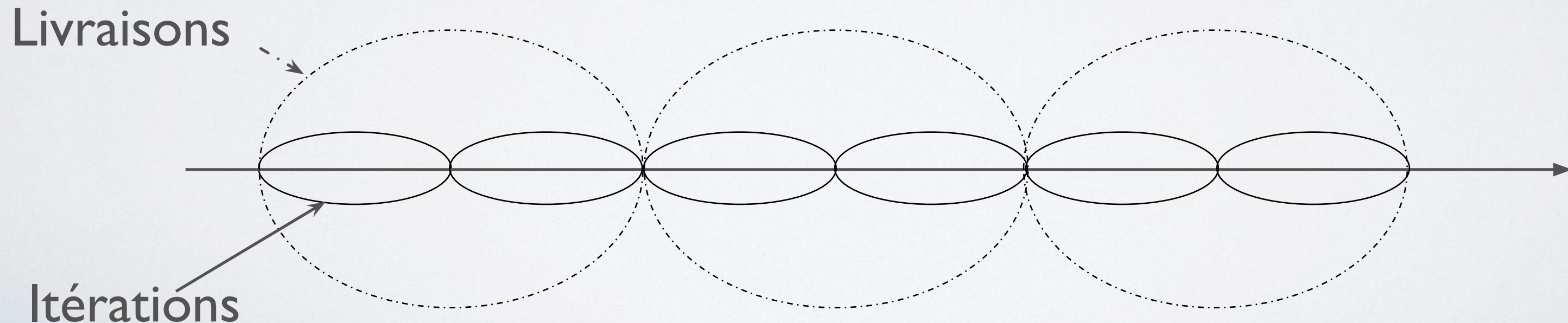
# PLANIFICATION ITÉRATIVE

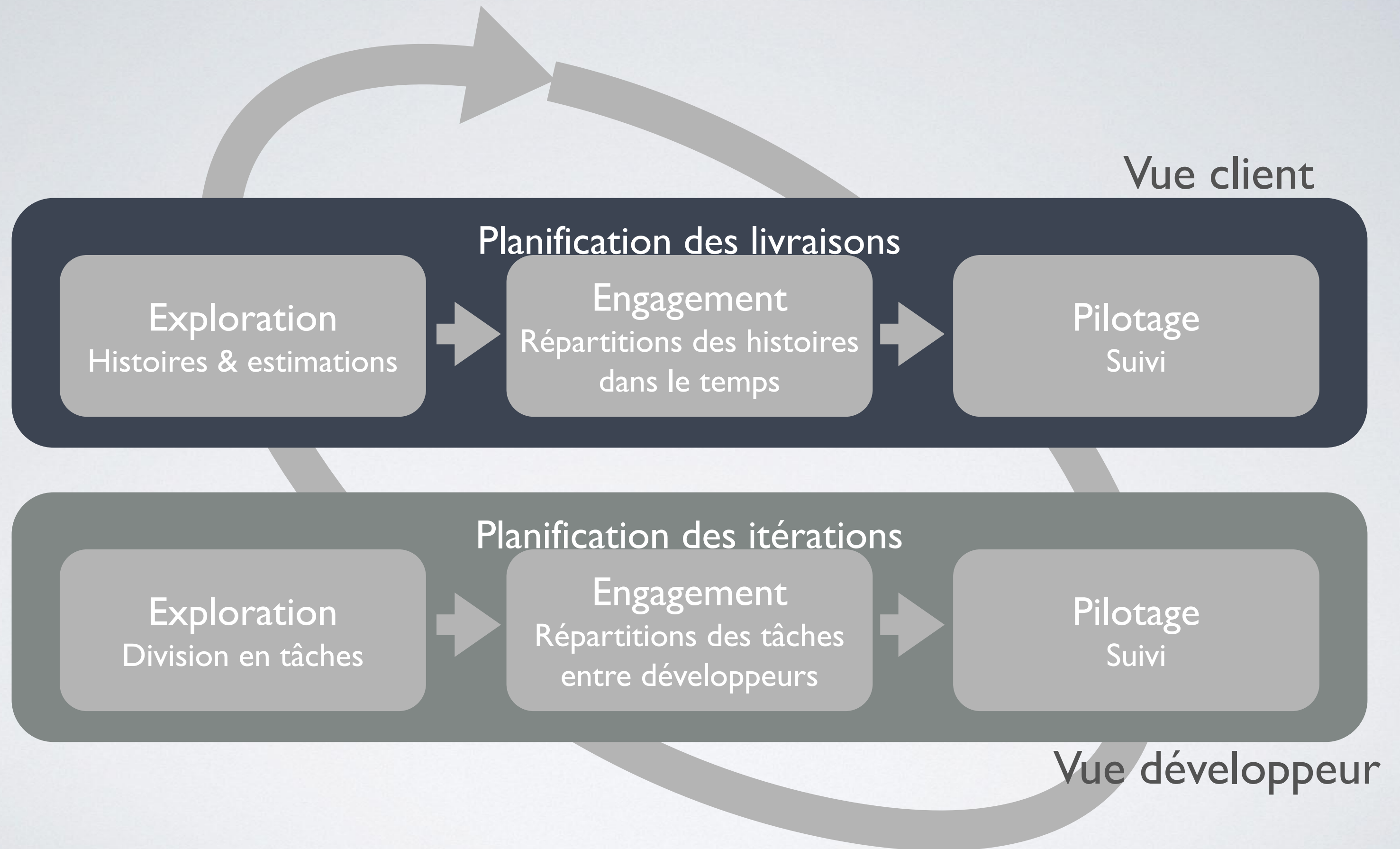
- Livraisons fréquentes définissent la répartition des livraisons, pas le contenu
- La planification itérative va définir le contenu des livraisons
- Séances de planification collectives (planning game)
  - Jeu = moins de tension
  - But = tirer le meilleur produit du projet



# PLANIFICATION ITÉRATIVE

- Ce jeu décompose le contenu du projet en une suite de *livraisons* et d'*itérations*
  - une itération doit durer entre 1 à 3 semaines
  - une livraison est un ensemble d'itérations

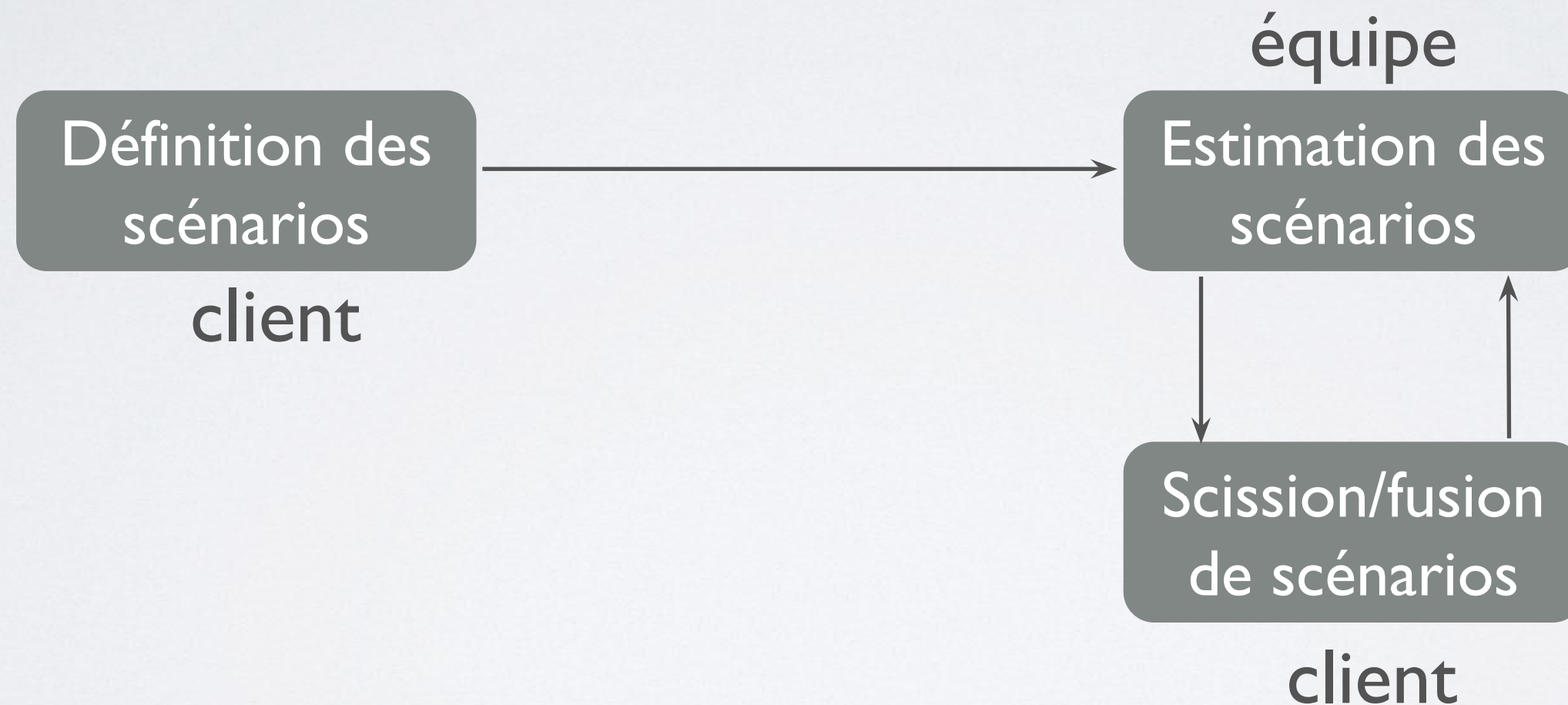






# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION



# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION

- Définition des scénarios client (user stories)
  - Histoire complète avec un début et une fin
  - Rédigés par le client lui-même, souvent avec le coach
  - A la main et dans son propre langage (simplicité)
  - Décrit **les interactions** entre l'utilisateur et le système pour une fonctionnalité donnée
  - Très sommaires, le plus simple possible



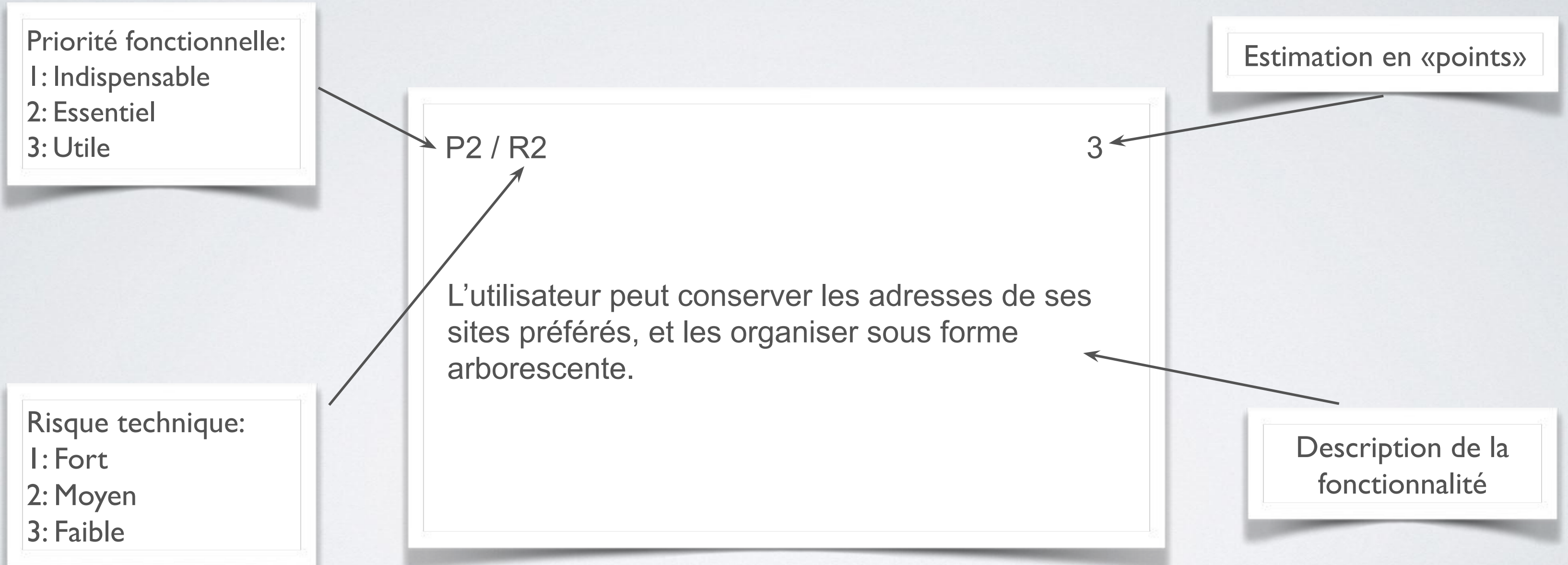
# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION

- Chaque scénario est noté sur une petite fiche cartonnée (A5)
  - Parfait pour la manipulation.
  - Un scénario inutile ? On déchire la fiche.
  - Il manque un scénario ? On sort une fiche vierge.
  - Peuvent être triées, partagées, échangées.
- => Toujours dans la valeur de *simplicité* d'XP

# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION





# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION

- Définition des scénarios client (user stories)
  - Parallèle avec Use Cases: les user stories ne définissent pas précisément une fonctionnalité
  - Ils servent juste à la planification
    - Les détails seront échangés entre le client et les développeurs au moment venu
    - **C'est une promesse d'une discussion à venir**

# CLIENT PLANNING

- A chaque scénario, le client en évaluera la priorité (P1 indispensable, P2 essentiel, P3 utile).
- Après ce processus d'écriture et de compréhension, une première séance de planification (*planning game*) se tient. Le premier plan de livraison en sortira.
- En concertation avec les programmeurs, le client répartira ses scénarios dans les itérations pour avoir les plus importantes d'abord.



# CLIENT TESTS DE RECETTE

- Le client devra définir les tests de recette pour les scénarios de l'itération
- Il les réalise avec l'aide du testeur

# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION

- Définition des scénarios client (user stories)
  - Élément de planification => facile à estimer et à suivre
    - granularité fine pour avoir une vision assez complète et en implémenter plusieurs en une livraison
      - ici max deux semaines de boulot par user story
  - doit être testable (tests de recette) prouvant que la fonctionnalité est finie



# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION

- Définition des scénarios client (user stories): exemples (pour un navigateur web)
  - «L'utilisateur peut conserver les adresses de ses sites préférés et les organiser sous forme d'arbre.»
  - «Au démarrage, afficher automatiquement la dernière page visitée par cet utilisateur.»
  - «Proposer un mode visualisation en plein écran.»
  - «Le temps de lancement de l'application doit être inférieur à 10 secondes sur la machine de référence.»

# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION

- Client aidé par les programmeurs le plus vite possible
  - N'écrire que les titres
  - Raconter oralement chaque histoire
  - Les programmeurs peuvent se faire une idée générale de l'application
  - Ensuite, le client peut en définir une dizaine (ceux les plus clairs pour lui)
  - Le client lit alors chaque scénario aux programmeurs, qui lui posent toutes les questions nécessaires à leur compréhension
  - Le client, via ce feedback, corrige et apprend son «métier de client»
  - Avec plus d'assurance, le client peut définir les scénarios les plus importants pour lui.



# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION

- Estimation des scénarios
  - Les développeurs essaient de déterminer l'envergure et la difficulté => estimation fiable
    - L'estimation doit intégrer la production du code, de la documentation, des tests unitaires et de recette
    - L'estimation se fait en points (pas en jours pour que ce soit plus simple) en comparant les scénarios entre eux.
    - Premières estimations: 1 point = 1 semaine de temps idéal (dans le cadre du projet: une heure de temps idéal)

# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION

- Lever les inconnues techniques
  - si trop d'inconnues techniques pour estimer, arrêt temporaire de la séance d'estimation et prototypes rapides (*spikes*)
  - si points techniques à discuter (comme montée en charge), le faire entre développeurs (avec consultants s'il le faut)



# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION

- Scission/fusion de scénarios
  - Si granularité trop grosse (plusieurs semaines de travail), les développeurs demandent au client de scinder le scénario en plusieurs
  - Si plusieurs scénarios sont estimés à moins d'un point, une fusion permettra de mieux estimer le temps

# PLANIFICATION DES LIVRAISONS

## PHASE D'EXPLORATION

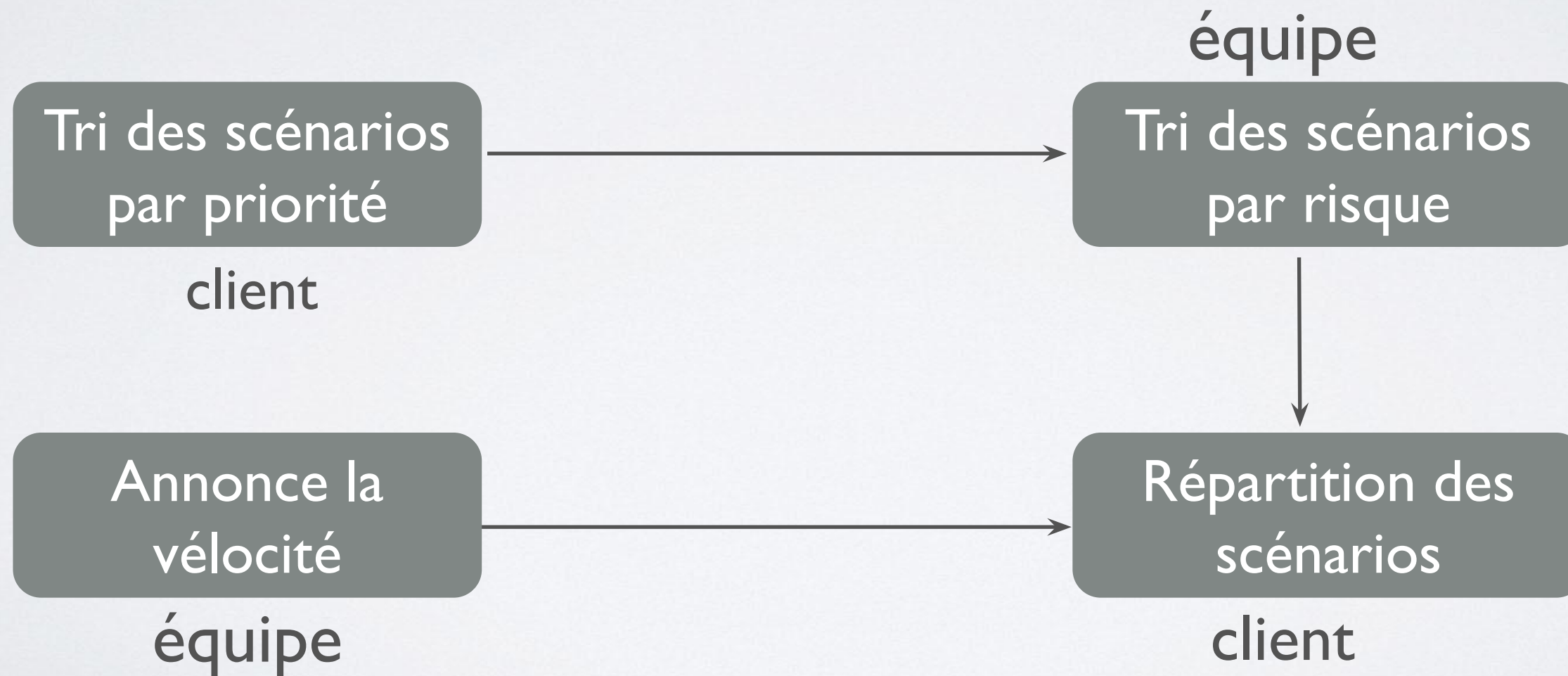
- Combien de temps dure cette phase ?
  - Jusqu'au moment où le client a fini de présenter tous ses besoins et que tous les scénarios sont estimés
  - Plusieurs heures/jours durant les premières itérations
  - Quelques heures, plus tard dans le projet
  - Dépendant de la vision à long terme (bien définir tous les besoins permet de mieux estimer le projet à long terme)



# PLANIFICATION DES LIVRAISONS

## PHASE D'ENGAGEMENT

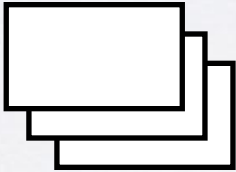
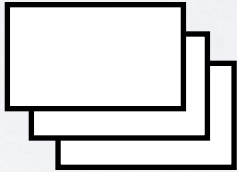
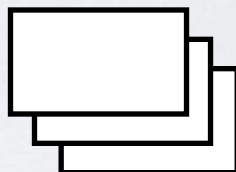
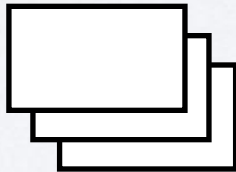
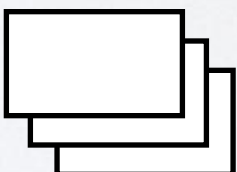

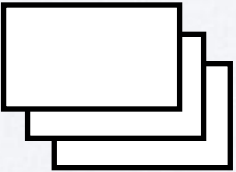


- But: Répartir les scénarios parmi les livraisons à venir pour établir le plan de livraison



# PLANIFICATION DES LIVRAISONS

## PHASE D'ENGAGEMENT

Tris

Risque Priorité	Fort	Moyen	Faible
Indispensable			
Essentiel			
Utile			



# PLANIFICATION DES LIVRAISONS

## PHASE D'ENGAGEMENT

- Annonce de la vélocité de l'équipe
  - La vélocité correspond au total des estimations en points des scénarios qui ont été totalement et correctement implémentés au cours de l'itération précédente.
  - Une fois que le client en a connaissance, il sait combien de points il peut «dépenser» à chaque itération pour «acheter» des scénarios
  - Pour la première itération: le coach fournit sa propre estimation de la vélocité
    - $\# \text{ binômes} * \# \text{ semaines/itération} * \% \text{ réel du temps idéal} * \# \text{ pts par semaine de temps idéal}$

# PLANIFICATION DES LIVRAISONS

## PHASE D'ENGAGEMENT

- Répartition des scénarios
  - Le client choisit, parmi les 9 groupes, des scénarios dont la somme des coûts en points est égal à la vélocité de l'équipe et ce, livraison après livraison
  - Le client doit s'assurer que les scénarios les plus importants à ces yeux soient traités au plus tôt
  - Rôle du coach est très important: faire respecter l'équilibre client/développeurs

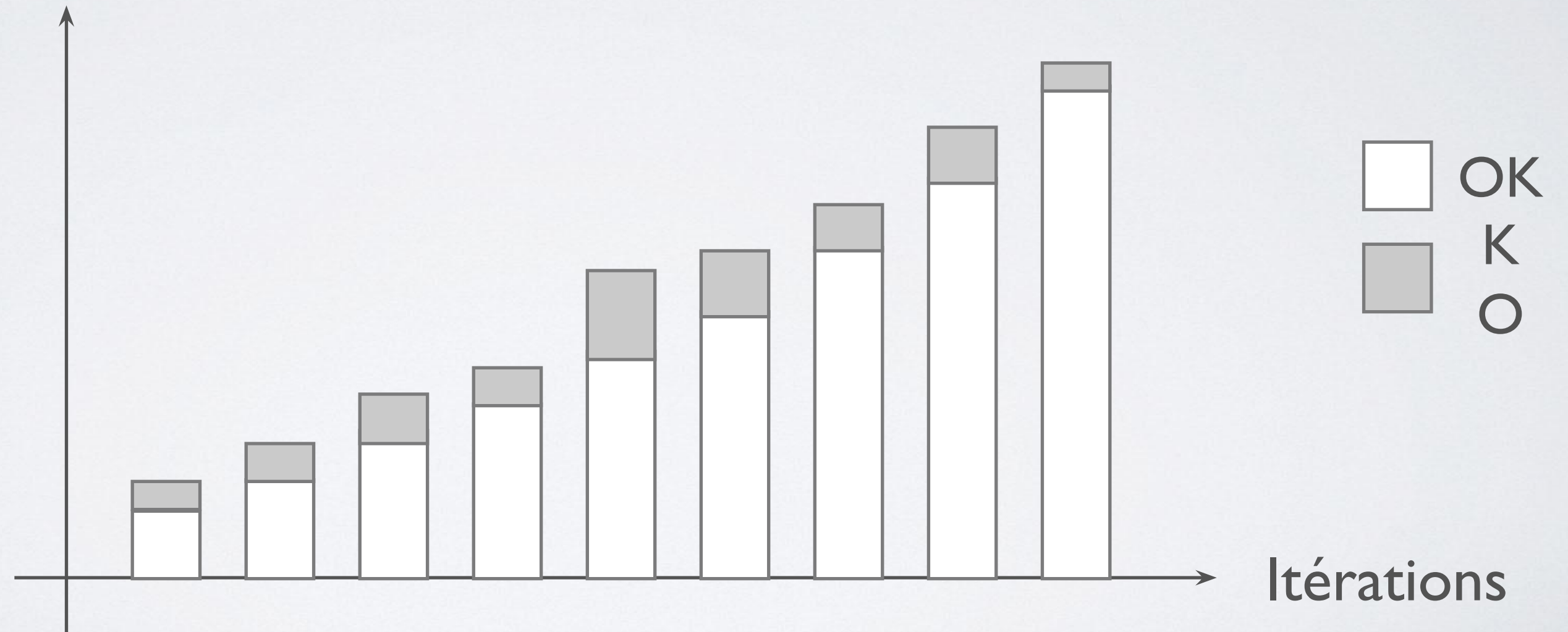


# PLANIFICATION DES LIVRAISONS

## PHASE DE PILOTAGE

Suivi des tests de recette

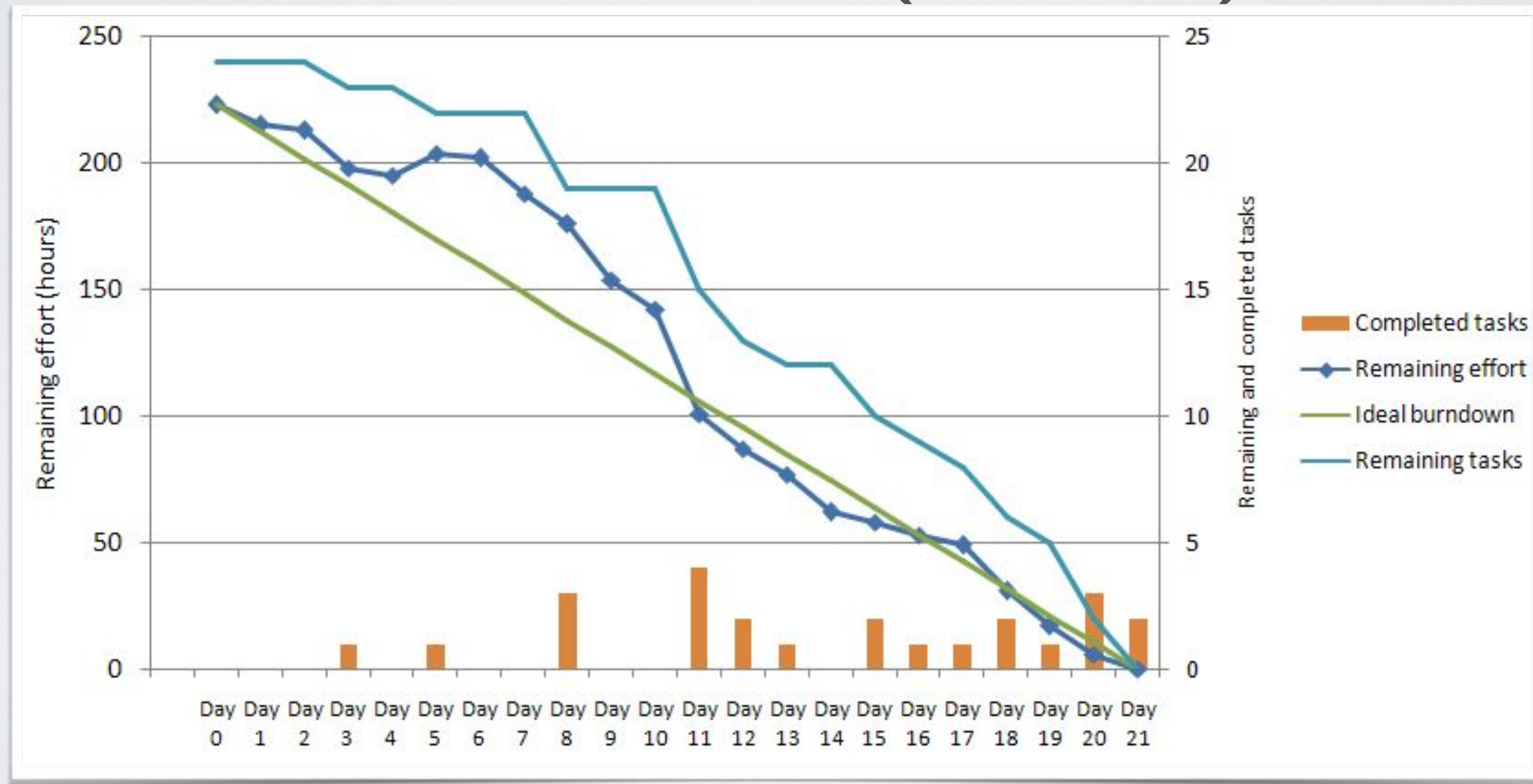
Nombre de tests



# PLANIFICATION DES LIVRAISONS

## PHASE DE PILOTAGE

### Burndown chart (! Scrum !)





# PLANIFICATION DES LIVRAISONS

## PHASE DE PILOTAGE

### Gestion des défauts et de leurs corrections

- Si un défaut apparaît:
  - soit écrire un scénario avec une estimation du coût de la réparation (toujours avec une importance maximale)
  - soit arrêter les tâches en cours et corriger le problème (normalement un défaut devrait être assez rare)

# PLANIFICATION DES LIVRAISONS

## FIN DE CYCLE

- Phase de pilotage s'arrête à la livraison
- Livraison effectuée même si tous les scénarios n'ont pas été implémentés
  - On joue sur la variable *contenu* et non *délais*
- Ensuite, il faut célébrer l'événement (repas, ...) !



# PLANIFICATION DES LIVRAISONS DÉBUT D'UN NOUVEAU CYCLE

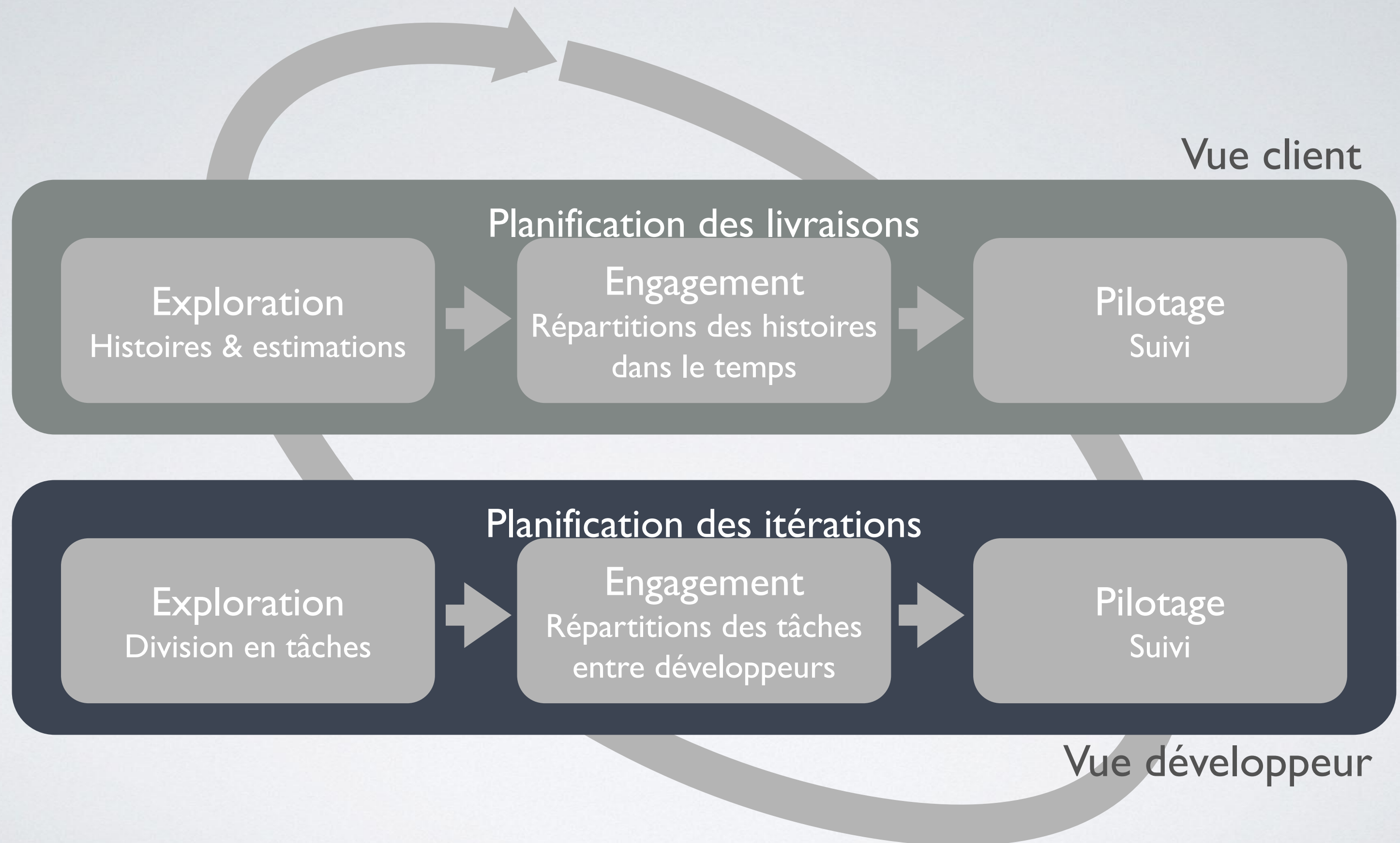
- Redéfinir les durées effectives de réalisation des scénarios
- Mettre à jour les durées des scénarios restants si nécessaire
- Déterminer avec plus de précision la vélocité

# PLANIFICATION DES LIVRAISONS

## FIN DU PROJET ?

- Décidée par le client
  - lorsque les scénarios restants ne valent plus l'investissement





# PLANIFICATION D'ITÉRATION

## PHASE D'EXPLORATION

- A chaque début d'itération
  - Les développeurs posent toutes les questions nécessaires au client
  - Division des scénarios en tâches techniques (en présence du client)
    - 1 tâche ~ 2 à 4 heures pour un binôme
  - Souvent, un peu de conception est nécessaire.



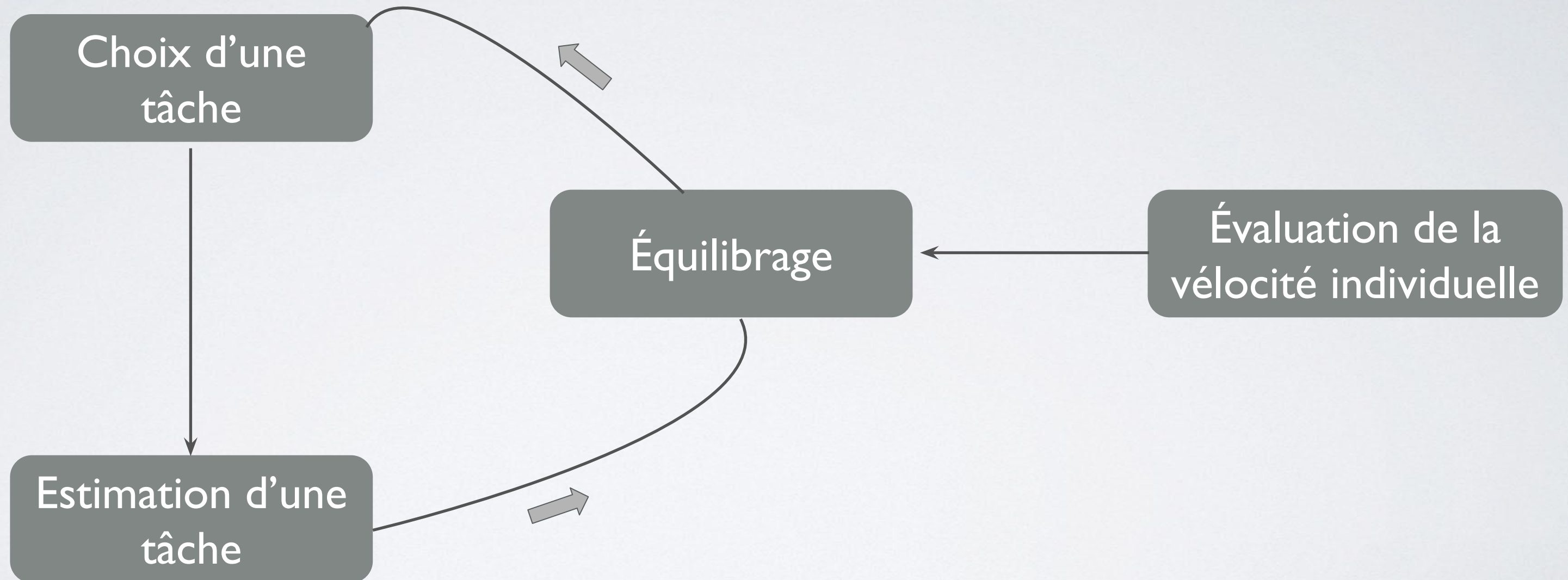
# PLANIFICATION D'ITÉRATION

## PHASE D'EXPLORATION

- Exemple
  - Scénario: «L'utilisateur peut conserver les adresses de ses sites préférés»
  - Tâches :
    - «Créer une classe SitePréféré»,
    - «Ajouter une option au menu Adresses»
    - ...

# PLANIFICATION D'ITÉRATION

## PHASE D'ENGAGEMENT





# PLANIFICATION D'ITÉRATION

## PHASE D'ENGAGEMENT

- Choix et estimation d'une tâche
  - Développeur volontaire
    - Il devient responsable de cette tâche
    - Il estime le temps en heures de «temps idéal»
      - en comparaison avec les tâches déjà effectuées ou son expérience pour les premières tâches

# PLANIFICATION D'ITÉRATION

## PHASE D'ENGAGEMENT

- Un tableau de la répartition des tâches est créé

Finie ?	Tâche	Qui	Prévu
	Sauvegarde/chargement des préférences utilisateur	FP	2
	Impression des raccourcis clavier (commande dans menu)	EL	2
	...	...	...



# PLANIFICATION D'ITÉRATION

## PHASE D'ENGAGEMENT

- Équilibrage des charges
  - La charge doit être uniformément répartie
  - Les plus chargés donnent aux moins chargés
  - Attention: travail en binôme => 50% de temps pour les tâches personnelles

# PLANIFICATION D'ITÉRATION

## PHASE D'ENGAGEMENT

- Ajout/Suppression de scénarios client
  - Après la répartition des tâches, si il reste des tâches à répartir ou si certains programmeurs sont sous-chargés
    - Demander au client d'ajouter/supprimer des scénarios



# PLANIFICATION D'ITÉRATION

## PHASE DE PILOTAGE

- Pour le côté pratique, voir le chapitre sur la collaboration.
- Deux fois par semaine:
  - le tracker fait le tour des programmeurs:
    - «Combien de temps as-tu passé sur chaque tâche ? »
    - «Combien de temps penses-tu encore passer sur chaque tâche ? »

# PLANIFICATION D'ITÉRATION

## PHASE DE PILOTAGE

- Si détection d'un dérapage sur le planning:
  - Brainstorming
    - Solution plus simple ?
    - Binôme plus expérimenté
    - Changer de programmeurs responsables pour la tâche
  - Si toute l'équipe chargée, demander au client de retirer un scénario



# PLANIFICATION D'ITÉRATION

## PHASE DE PILOTAGE

- *Stand-up meetings*
  - 1x par jour (souvent le matin)
  - 10 minutes environ
  - debout ! (pour que ça aille vite)
  - Chaque développeur fait un point rapide sur sa situation et les difficultés rencontrées
    - Exprime ses besoins (binôme particulier, petite réunion de conception, obtenir de l'info du coach ou du client,...)

**Ce n'est pas une réunion technique !  
Ce n'est pas un suivi des tâches (cfr tracker) !**

# PLANIFICATION D'ITÉRATION

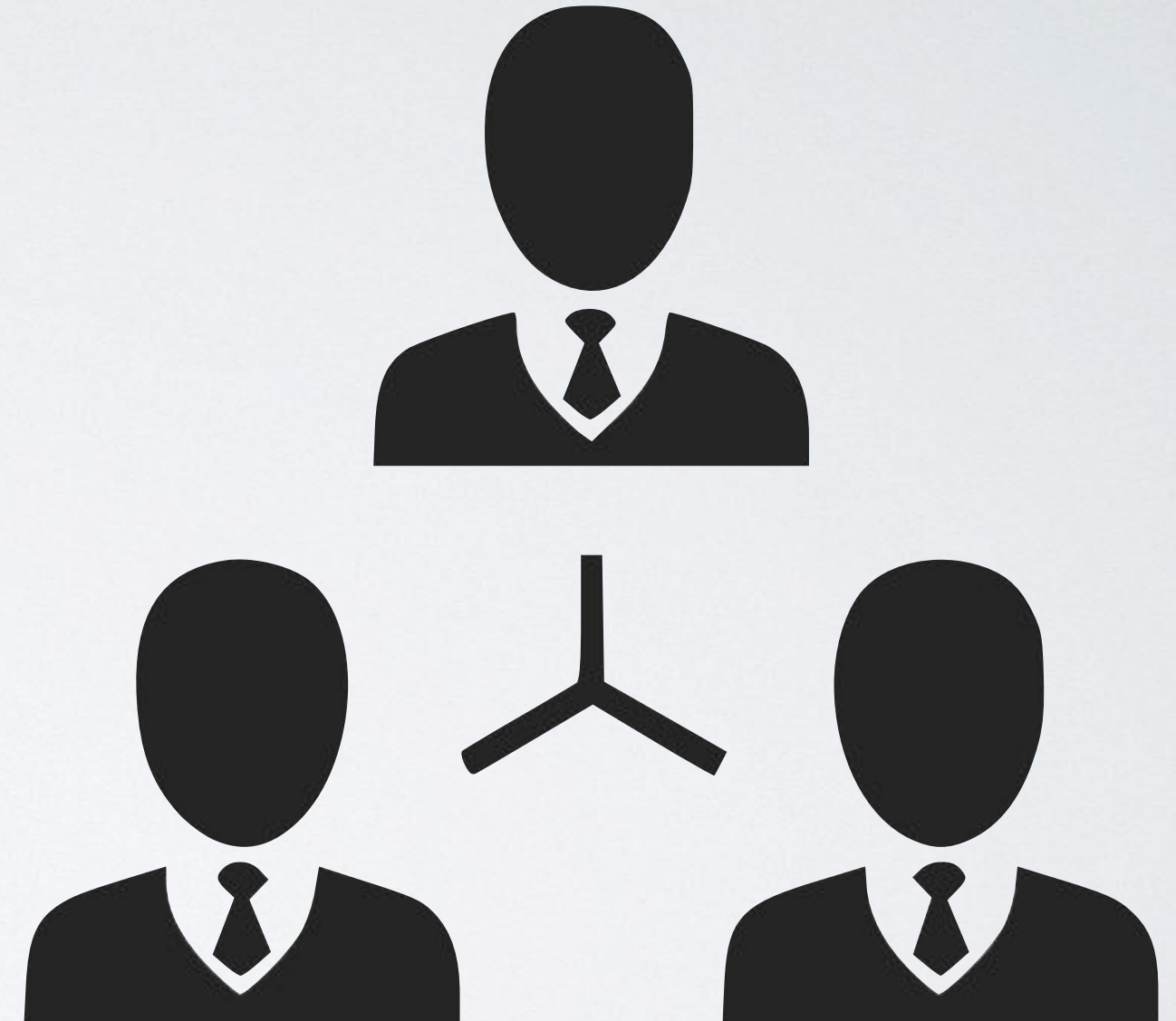
## FIN DE L'ITÉRATION

- Mini-livraison du logiciel au client
  - S'assure que tous les tests de recette passent
- Mise à jour des coûts des scénarios réalisés (somme des durées des tâches)
- On fête ça ! Boissons et gâteaux :-)



# CLIENT SUR SITE

- Comment favoriser la communication entre les programmeurs et le client ?
  - En les rapprochant géographiquement !
  - Dans le même bâtiment, voire le même bureau
  - Favorise les questions directes... et les réponses directes !



# CLIENT SUR SITE

- Le client apporte ses compétences métier
- Pas de document de spécifications => communication permanente avec le client
- Question d'ordre fonctionnel ?
  - méthodologie en cascade : le développeur doit trouver une solution (qui n'est peut-être pas celle voulue par le client)
  - XP: le client est là pour répondre à ce type de question, les développeurs ne sont pas là pour ça



# CLIENT SUR SITE

- Les tests unitaires valident la mécanique interne de l'application
- XP emploie les tests de recette (tests fonctionnels ou acceptance tests)
  - le client doit définir ces tests ([exemple](#))
  - tests automatiques également ([exemple](#))
  - équivalent à la spécification du produit

# CLIENT SUR SITE

- Pas évident d'avoir toujours quelqu'un disponible
  - Plusieurs personnes peuvent jouer le rôle du client
- Limiter au maximum le nombre de personnes éloignées du cadre final



# RYTHME DURABLE

- Trouver un rythme optimal pour l'équipe
  - Avoir du temps pour réaliser un travail de qualité... et de se reposer !
  - Il faut beaucoup d'énergie pour concevoir simplement, coder proprement, tester correctement et explorer tous les problèmes



# RYTHME DURABLE

- Si il y a surcharge de travail, il y aura perte de qualité
  - XP: pas d'heures supplémentaires 2 semaines de suite
- Si on n'y arrive pas, cherchez la vraie cause...





DES QUESTIONS ?