

Approche Agile

Les méthodes agiles

- 
- Cycle de développement **itératif**, **incrémental** et **adaptatif**
 - En 2001, est né le **manifeste Agile - adaptatif**
<http://www.agilemanifesto.org/>

Définition Agilité

L'agilité est la capacité à favoriser le changement et à y répondre en vue de de s'adapter au mieux à un environnement turbulent.

Aubry, C. (2014). Scrum, Le guide pratique de la méthode agile la plus populaire. Dunod, p3.

Manifeste pour les développements Agile

Nous
par la pratique et en aidant les autres à le faire,
reconnaissons la valeur des seconds éléments,
mais privilégions les premiers.

Les individus et leurs interactions plutôt que les processus et les outils

Des logiciels opérationnels plutôt qu'une documentation exhaustive

La collaboration avec les clients plutôt que la négociation contractuelle

L'adaptation au changement plutôt que le suivi d'un plan

Equipe

Les individus et leurs interactions plutôt que les processus et les outils

- Equipe de développeurs soudée et qui communique
- Equipe de développeurs de différents niveaux qui communiquent bien entre eux plutôt qu'une équipe composée d'experts fonctionnant chacun de manière isolée.

→ Communication

Application fonctionnelle

Des logiciels opérationnels plutôt qu'une documentation exhaustive

- D'abord un logiciel fonctionnel
- Importance du logiciel documenté (code) mais pas les documents produits relatifs au projet
 - **Logiciel testé ET documenté pour pouvoir le maintenir**
 - A minimiser : documents projet : rapport d'avancement, notes explicatives, planning revu et maintenu à jour en permanence, flux pour signature...

Collaboration

La collaboration avec les clients plutôt que la négociation contractuelle

- Le client doit être impliqué dans le développement
- On ne peut négliger les demandes du client
- Le client doit collaborer avec l'équipe et fournir un feedback continu sur le logiciel

→ logiciel adapté aux attentes du client.

Acceptation du changement

L'adaptation au changement plutôt que le suivi d'un plan.

- Réagir aux demandes de changement
- Planification flexible
 - Plan de développement au début du projet
 - Revu et remanié à chaque nouvelle itération :
 - Permettre l'évolution de la demande du client tout au long du projet
 - Prendre en compte les demandes d'évolution provoquées par les premières *releases* du logiciel.

Principes Agile (1)

- **Satisfaire le client** : plus haute priorité
 - Livrer **rapidement** des fonctionnalités à **grande valeur ajoutée**
- **Accueillir les changements** de besoins, même tard dans le projet pour donner un avantage compétitif au client
 - Conception orientée évolution.
- **Livrer fréquemment** un logiciel opérationnel avec des cycles de quelques semaines à quelques mois.
 - Mise en production rapide d'une version minimale du logiciel & ensuite nouvelles livraisons incrémentales.

Principes Agile (2)

- **Collaborer quotidiennement** entre utilisateurs (ou de leurs représentants) et développeurs.
- Réaliser les projets avec des personnes motivées, en fournissant l'environnement et le soutien dont elles ont besoin et en leur faisant confiance
- Communiquer par des **conversations en face à face**
- **Mesurer l'avancement par un logiciel opérationnel**
- Garder un **rythme de développement soutenable** : maintenir indéfiniment un rythme constant.



Principes Agile (3)

- Rechercher **l'excellence technique et une bonne conception.**
- Rechercher la **simplicité** - càd minimiser la quantité de travail inutile.
- Laisser l'équipe **s'auto-organiser**, ne pas imposer de processus.
- Laisser l'équipe réfléchir aux moyens de devenir plus efficace & adapter son comportement.

Agile : adaptatif

Planification (1)

- **Planification itérative.**
- Exemple : le développement d'un projet est prévu sur 3 mois, itération de 2 semaines. (14 US, 4 T)

Planning Projet												
Spécifications techniques et fonctionnelles	US1, US7, US8, T1											
			US5, US6, US4, T2									
					T3, T4, US2							
							US9, US11, US12, US3					
								US10, US13				
										US14, US15		
Semaines	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
	Itération 1		Itération 2		Itération 3		Itération 4		Itération 5		Itération 6	

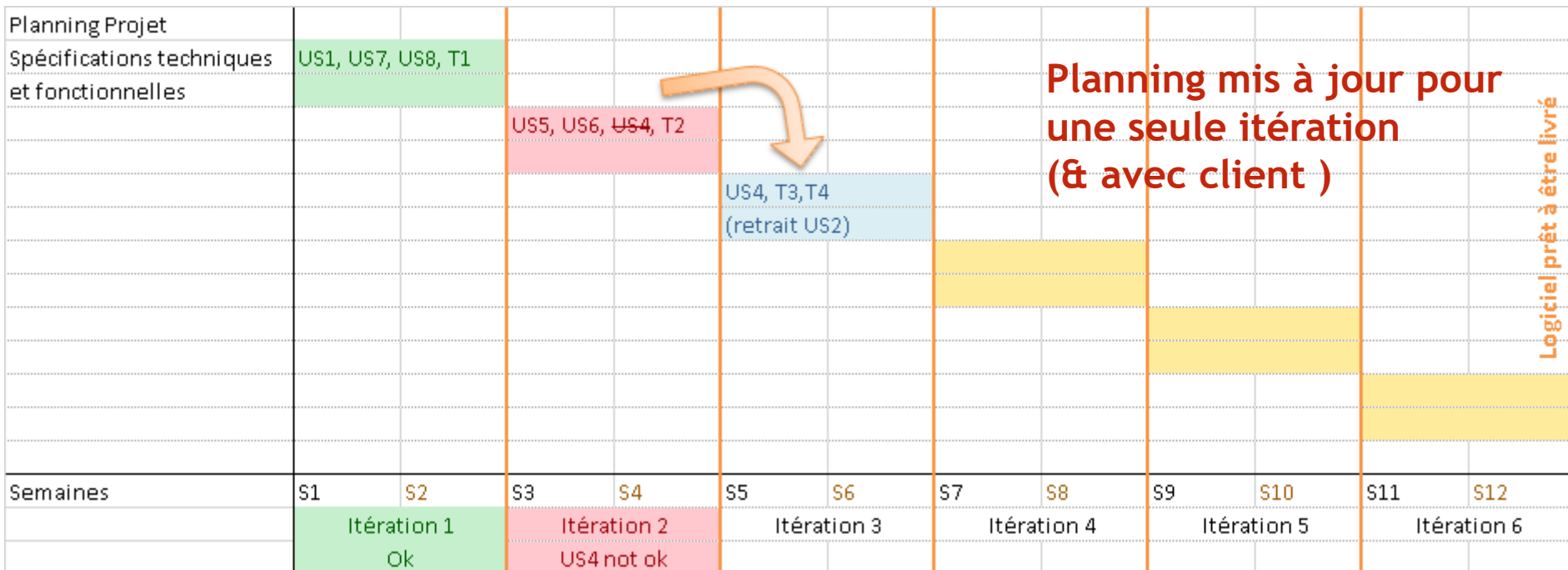
Planning initial : pas détaillé, on vérifie rapidement que l'on peut tout développer

Logiciel prêt à être livré

Agile : adaptatif

Planification (2)

- A la fin de chaque itération, on regarde ce que l'on a réalisé et on adapte le planning.
- Si ok en itération 1, mais US4 non développé en it2.



Agile : adaptatif

- **Fonctionnellement**, par adaptation systématique du produit aux changements de besoin.
 - La demande de changement est bienvenue mais
 - Changement n'est pas permanent.
 - Ne peut y avoir d'interruption intempestive du développeur qui travaille.
 - Engagement pour l'itération en cours n'est pas modifié.
 - On passe par une gestion des priorités et la demande de changement peut être différée.

Agile : adaptatif

- **Techniquement**, par remaniement régulier du code : **refactoring**.
 - Refondre le code source pour en améliorer la qualité.
 - Changer le design sans changer les fonctionnalités.
 - Fort encouragé dans les méthodes Agiles.
 - Outil de changement.

Deux méthodes agiles

eXtreme Programming

Scrum

2 méthodes

- eXtreme Programming
 - Premier projet en 1996
 - En perte de vitesse.
 - A permis de développer des concepts et des pratiques qui sont utilisés dans les méthodes Agile.
- Scrum
 - S'est imposé.

- <http://www.extremeprogramming.org/>
- <http://xprogramming.com/index.php>
- <http://extremeprogramming.free.fr/>

Concepts et pratiques XP

eXtreme Programming

Pratiques de programmation

- **Conception simple**
- **Refactoring**
 - Code en permanence remanié pour rester simple et compréhensible
- **Tests de non régression**
 - Batteries de tests de non-régression qui permettent de faire face aux modifications permanentes
- **Tests de vérification / d'acceptation**
 - Batteries de tests automatisées.

Pratiques de collaboration

- **Programmation en binôme**
 - Revue de code permanente
 - Apprentissage et partage des connaissances
- **Responsabilité collective du code**
 - Développeur responsable d'une partie du code
 - Tous responsables du code, chacun y ayant accès
- **Intégration continue**
 - Intégration des nouveaux développements chaque jour.

XP life cycle

User stories
Evaluation -> difficultés
Planning imprécis
mais revu de man.
régulière

Spécifications
de l'itération

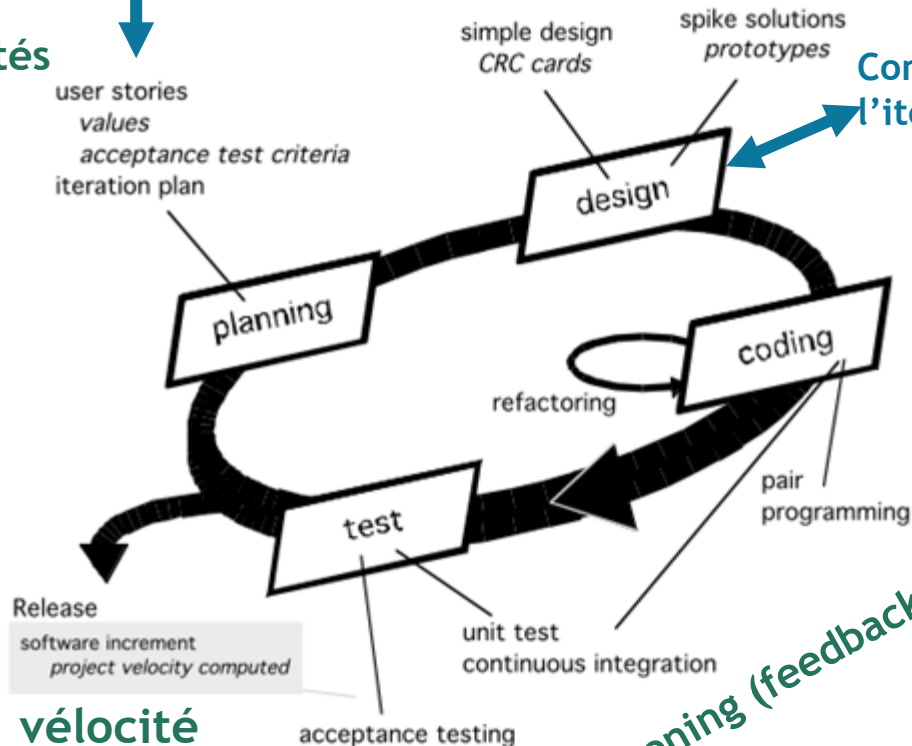


user stories
values
acceptance test criteria
iteration plan

simple design
CRC cards

spike solutions
prototypes

Conception de
l'itération



Itération :
+/- 2 semaines

Calcul du
temps de
travail

vélocité

Listening (feedback)

<http://top-ilmu.blogspot.be/2013/10/metode-agile.html>

03/05/2020

User stories

- Format associé aux méthodes agiles
- Description brève d'une fonctionnalité telle que vue par l'utilisateur
- Format écrit **court**, laissant de la place à la discussion orale
- Emergence rapide dans des ateliers collaboratifs
- Grande simplicité et donc grande lisibilité
- Histoire implémentée en une seule itération! (découpe si besoin)

Une user story est utilisée en tant que spécifications mais également pour l'estimation du temps de développement et la planification.

User stories : si Projet AE Agile

ID

102

Nom

Créer un nouveau client

Histoire

- **En tant que** patron
Je veux créer un client
Afin de pouvoir introduire son devis.

Validation

- Un client a un nom, un prénom, une adresse complète (rue, no, boîte, code postal, ville), un téléphone et un email.
- Toutes les données sont obligatoires.
- Un client peut être lié à un utilisateur de même nom, prénom, email et ville.

Poids

- (Effort estimé) - au moment de la création de la US, parfois estimation ou laissé vide et sera apprécié au moment du planning de l'itération

ID

103

Nom

User stories : si Projet AE Agile

Introduire un devis

Histoire

- **En tant que** patron
Je veux introduire un devis et son client
Afin d'assurer le suivi des travaux, du devis jusqu'à la facture finale.

Validation

- Un devis est lié à un client, a une date, un montant total et une durée.
- Il peut avoir des photos du jardin avant travaux.
- Il a un ou plusieurs types d'aménagements.

Poids

- (Effort estimé) 3

Questions ?

Vélocité

Début itération

- Chaque US a un poids
- On planifie un nombre d'US équivalent à notre vélocité.

Fin d'itération

- Somme des poids des US **effectivement réalisées** = vélocité de l'itération.



Vélocité : exemple

Début itération

- User Story A : 2 points
- US B : 3 points
- US C : 3 points

Fin d'itération

- US A : terminée
- US B : non terminée
- US C : terminée

➔ **Vélocité = 5 points.**

Prévoir
itération
suivante :
5 points



La vélocité est une **constatation à posteriori**.

Présentation de l'entreprise

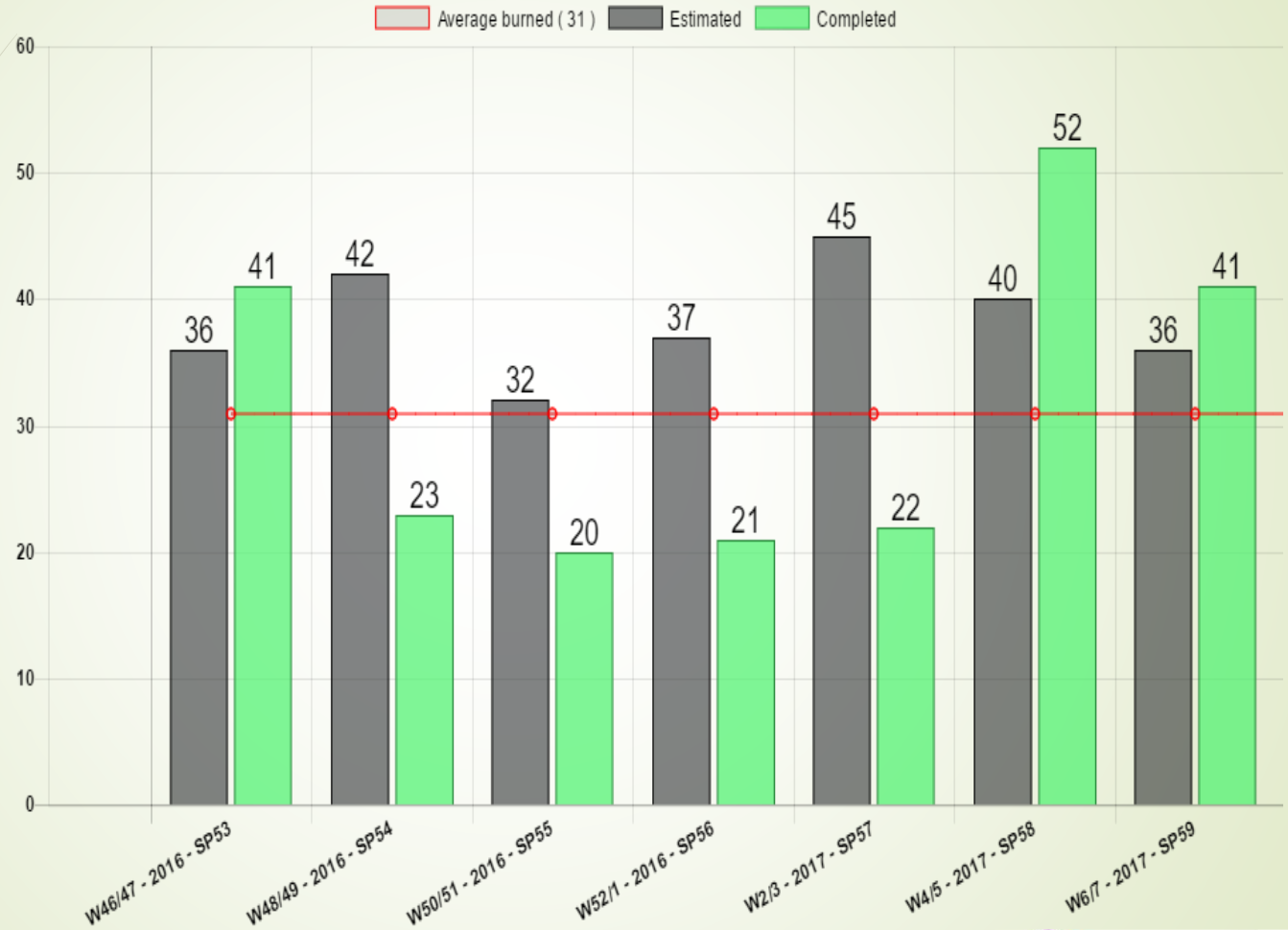
- L'entreprise
- Les "work agreements"
- Les produits

**Stage
d'observation de
Benjamin Bergé
2017**

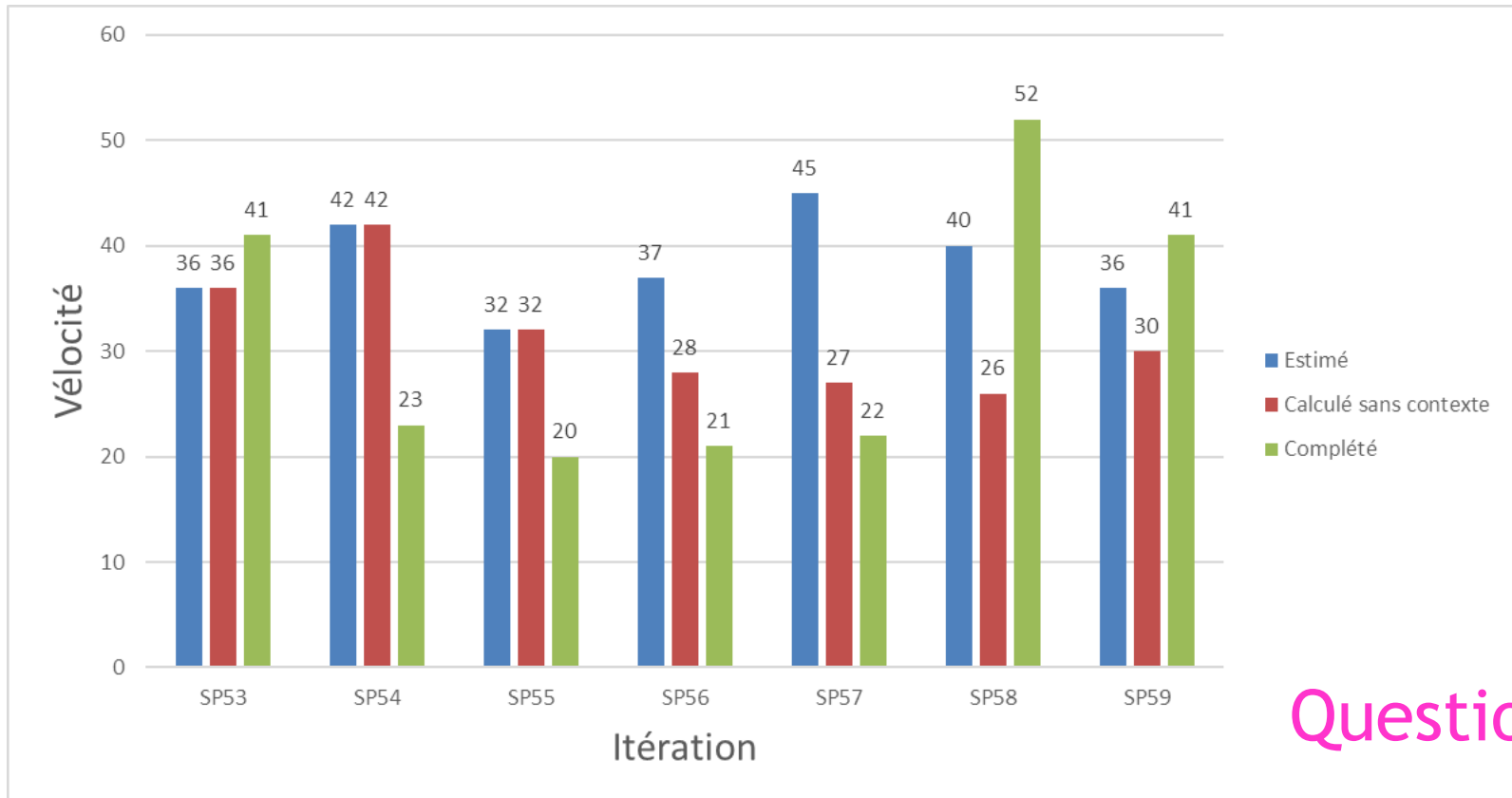
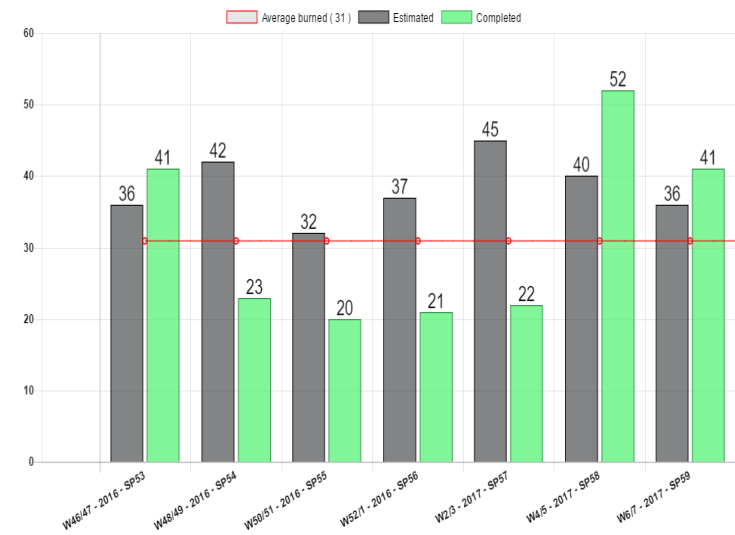
WORK AGREEMENTS

- Alert the team quicker when you need help.
- Perform pair programming on difficult issues when somebody isn't familiar with a part of the development.
- QA should be done faster and set in the agenda.
- Better meeting objectives at start of the meeting.
- Be aware before doing a "push-f".
- When objectives of a feature is clear -> don't lose focus on it.
- When working remotely be available.
- Be open minded for others opinions (work related or not)
- Internal documentation has to be regularly updated!

Organisation de l'équipe de programmeurs



Commentaire Mme Lehmann :
Calcul vélocité (pas de contexte) -
équipe stable - quantité travail
constante



Questions ?

<http://www.scrumalliance.org>
<http://www.scrum.org/>

Scrum

An Agile framework

Scrum

Méthode Agile

- Apporter plus de valeur aux clients et aux utilisateurs
 - Maximiser la valeur ajoutée
 - En réalisant d'abord les fonctions à plus haute valeur ajoutée
 - En changeant les priorités et même les fonctions
 - ➔ Voir définition du sprint backlog
- Apporter une plus grande satisfaction dans le travail
 - Equipe auto-organisée.

Scrum

Scrum fournit un **cadre** pour le développement d'un produit complexe.

Scrum est adaptatif :

- Idée : il est impossible de définir tout dès le début : les spécifications peuvent changer, des outils ou technologies inconnus entreront en jeu, etc..
- pour s'adapter aux changements, les travaux à faire sont ajustés à la fin de chaque itération.

Cadre

Cadre très léger

- 3 rôles
- Itérations
- Réunions au début et à la fin de chaque itération
- Mêlée quotidienne
- Backlog de produit
- Peu de production documentaire.

Rôles

1. Product owner

- **Product Owner : directeur de produit**
 - Représentant du client et des utilisateurs dans l'équipe
 - Représentant ayant la vision du produit et l'autorité pour donner les priorités aux requirements du client
 - Responsable du **Product Backlog**
 - Responsable du résultat auprès des autres parties
 - Canal de communication avec les parties prenantes.

Rôles

2. Scrum master

- **Scrum Master : gestionnaire**
 - Il **facilite** l'application de Scrum dans l'équipe
 - Il s'assure que la méthodologie soit respectée et provoque les changements organisationnels nécessaires à cela
 - Il supprime ce qui pourrait interrompre les équipe et les protège des interférences extérieures
 - Il s'assure de l'amélioration des pratiques et de l'organisation du travail
 - C'est le **coach** de l'équipe.

Rôles

3. l'équipe de développement

- **Equipe :**
 - Ensemble des individus participant aux activités de développement,
 - Individus ayant des compétences transversales
 - Equipe performante et **autoorganisée**
 - Equipe centrée sur les livraisons.

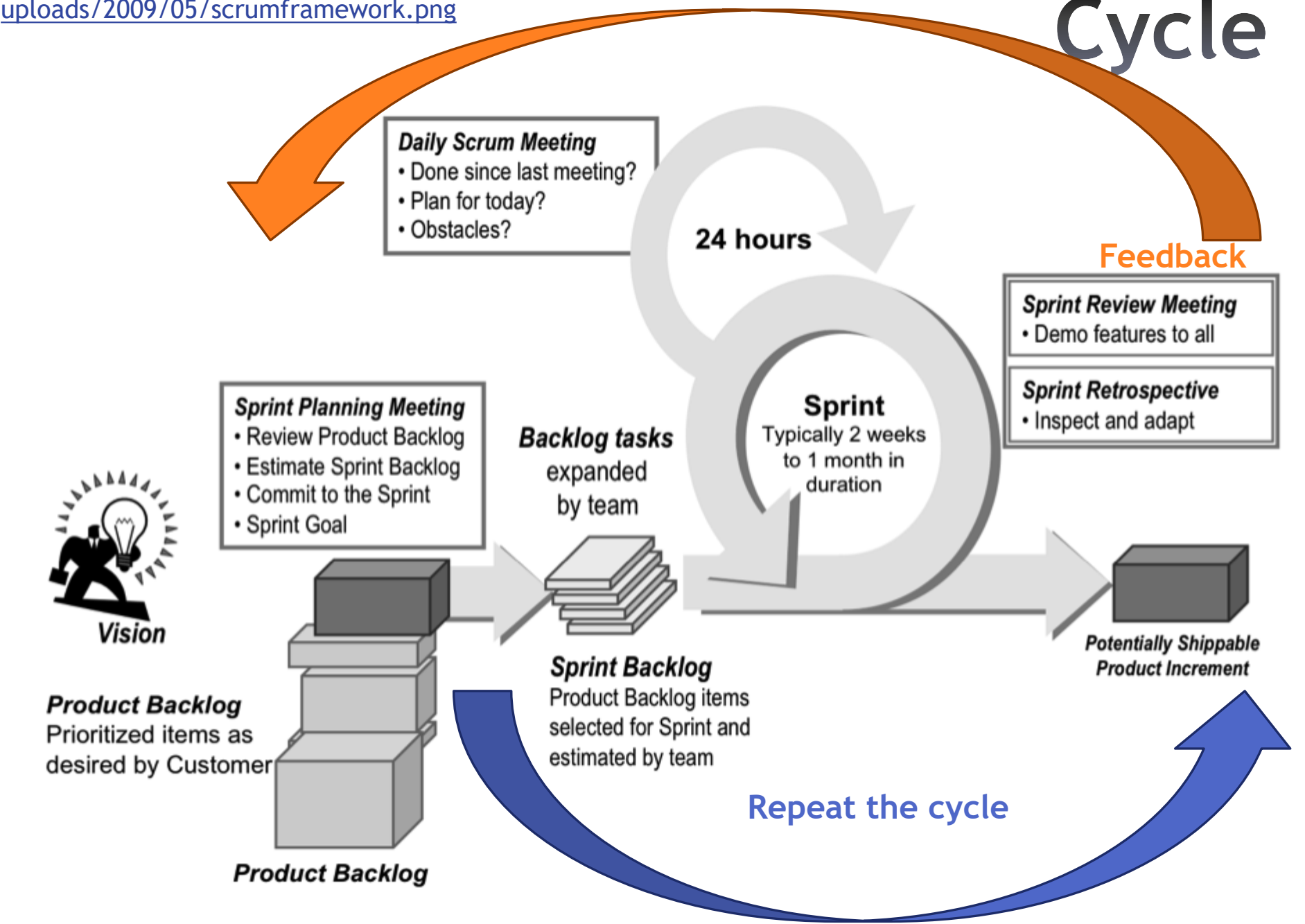
Concepts

- **Sprint : unité de temps** qui permet de rythmer les développements, correspondant à une **itération**
 - Sprint de durée courte (maximum un mois)
 - Pas de chevauchement
 - Rythme régulier : tous les sprints de même durée
 - Pas de changement de date de fin si le développement n'est pas fini
 - Fixation de la quantité de travail à produire en fonction de la durée et de la taille de l'équipe
 - ➔ budget fixe (si la taille de l'équipe est stable mais ceci est recommandé).

Concepts

- **Product Backlog** : liste, ordonnée par priorité, des fonctionnalités pour le produit, définie par le directeur de produit
 - Les nouvelles demandes ou corrections de bugs sont ajoutées dans le product backlog.
- **Sprint Backlog** : liste des fonctionnalités qui sont développées dans le sprint
- **Version** : plusieurs sprints peuvent être nécessaires pour développer une version du produit.

Cycle



Réunion de début de sprint

(Sprint planning)

Planification du Sprint :

- Direction : ScrumMaster
- Définition de l'objectif du sprint (Sprint Goal)
- Analyse du haut de la liste du « ProductBacklog »
- Définition du « SprintBacklog » en fonction de la capacité de l'équipe et de la priorité des tâches.
 - Pour cela, il faut avoir estimé le poids des user stories
 - Complexité, longueur, risques de complication
 - Division éventuelle
 - Consensus sur le poids
 - Comparaison avec vélocité (moyenne de la vélocité des 3 derniers sprints, par exemple).

Mêlée journalière (1)

Scrum Meeting:

- **Objectif** : savoir ce que chacun fait, se synchroniser, mesurer l'avancement, s'entre-aider
 - **Rapide tour d'avancement, équipe debout**
 - Réunion une fois par jour à heure fixe
 - Mise en commun des apports de chacun
 - Partage des difficultés rencontrées.

Mêlée journalière (2)

Scrum Meeting (suite):

- **Réponse aux 3 questions :**
 - Qu'as-tu **terminé** depuis la précédente réunion?
 - Que penses-tu pouvoir **terminer** d'ici la prochaine réunion?
 - Quels **obstacles** rencontres-tu en ce moment? (risque de ne pas respecter le sprint backlog)
- **Ajustements possibles**
 - Scrum Master détermine l'avancement par rapport aux engagements. Si nécessaire, ajustements.

Réunion de revue

(sprint review)

Réunion de revue

- Démonstration des nouvelles fonctionnalités (en interne avec l'équipe)
- **Feedback** du directeur de produit
- Redéfinition des priorités du ProductBacklog
- Identification des user stories à traiter lors du prochain Sprint.

Rétrospective

Rétrospective

- Prise de recul sur l'itération qui s'est terminée
 - Identification des améliorations potentielles (processus, méthode de travail)
 - Détermination du « comment » réaliser ces améliorations
 - Ce que l'on aimerait mettre en place
 - Ce que l'on souhaite arrêter
 - Ce que l'on souhaite continuer.
- ➔ Cette rétrospective permet donc d'améliorer le fonctionnement de l'équipe et d'augmenter la satisfaction de ses membres.

Cycle & Sprint

A l'intérieur d'un Sprint

- développement d'un produit partiel :
Spécifier, analyser, concevoir, développer, tester, documenter et intégrer pour chaque US

Pendant le cycle de sprints

- Spécifications continues : le client précisera ses demandes au moment où elles sont développées
- Conception continue : l'architecture va évoluer pendant la vie du projet
- Tests dès le premier sprint.

Quand est-ce fini?

- Les Sprint se répètent jusqu'au moment où :
 - Il y a eu assez de fonctionnalités développées dans le ProductBacklog ou
 - Le budget est épuisé ou
 - La date limite du projet est atteinte.
- Les fonctionnalités à plus haute valeur ajoutée ont été développées !

Périodes

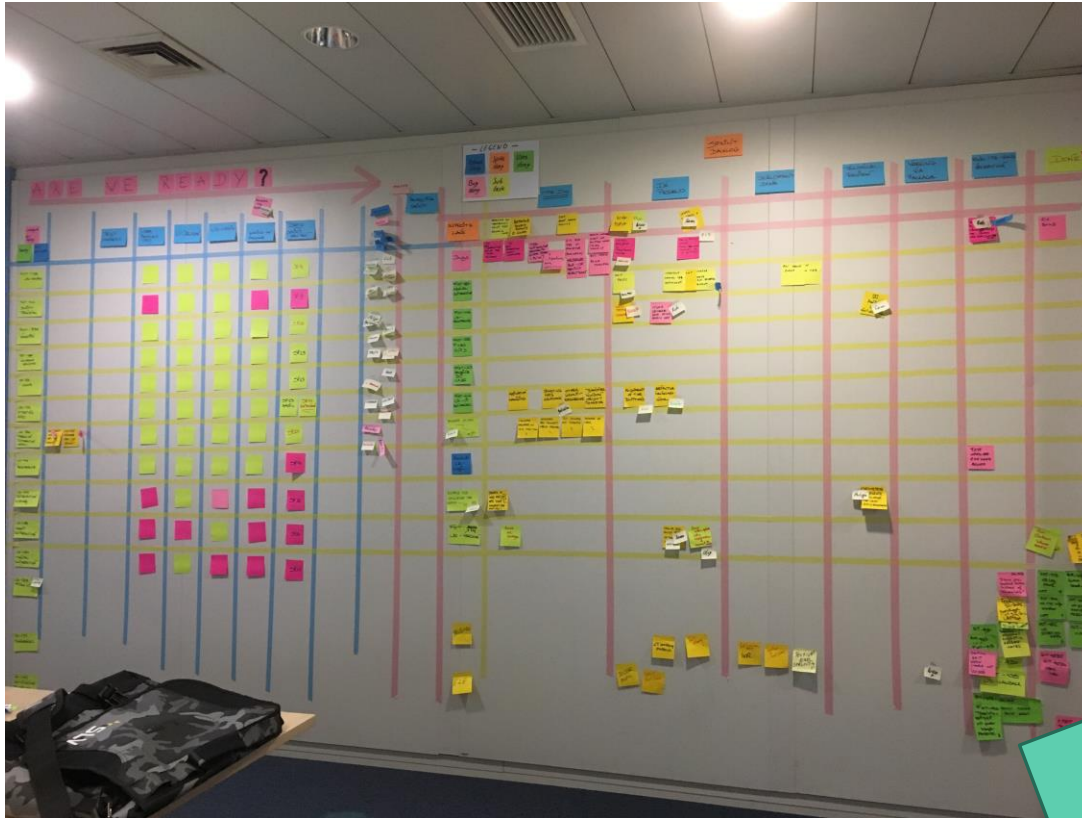
Il y a quand même 3 périodes en Scrum !

- Avant le premier sprint, après discussions avec le client et remise de prix
 - Définition backlog initial
 - Définition de la vision
 - Constitution de l'équipe, mise en place de l'environnement
 - Maîtrise de l'architecture
 - Première planification générale
- **Les sprints : période centrale**
- Une période de clôture: tests d'acceptation sur tout le produit/projet/release; déployer l'environnement de production, écrire les manuels et donner la formation.

Exemples

Au quotidien

Scrum board : répartition des tâches



ARHS
Developments
Belgium – client
Proximus

Stage observation
Ronsmans Thomas
2016-2017

Couleurs ont une
signification : par
exemple, les jaunes
représentent les
tâches du front-end

Stage d'observation de
Frédéric Hubert
2017

Scrum





Une illustration

Description de l'organisation du travail

Stand Up Meeting

Stand up meeting

- Tous les matins à 10h
- Chaque développeur présente un résumé des tâches qu'il a accomplies le jour précédent et formule les remarques qu'il a à faire sur celles-ci
- Le responsable répartit les nouvelles tâches
- Un dashboard est mis-à-jour .

	To Do	On Going	Done	Test	Prod
Legacy					
User story 1					
User story 2					
User story 3					
User story 3					
...					

Parking


Description de l'organisation du travail



Organisation en Sprints de 2 semaines

Sprint
2 semaines

Mercredi	Sélection des scénarii utilisateurs Développement
Jeudi	Développement
Vendredi	Tests
Lundi	Correction des bugs
Mardi	Développement
Mercredi	Développement
Jeudi	Développement
Vendredi	Tests
Lundi	Correction des bugs
Mardi	Mise en production

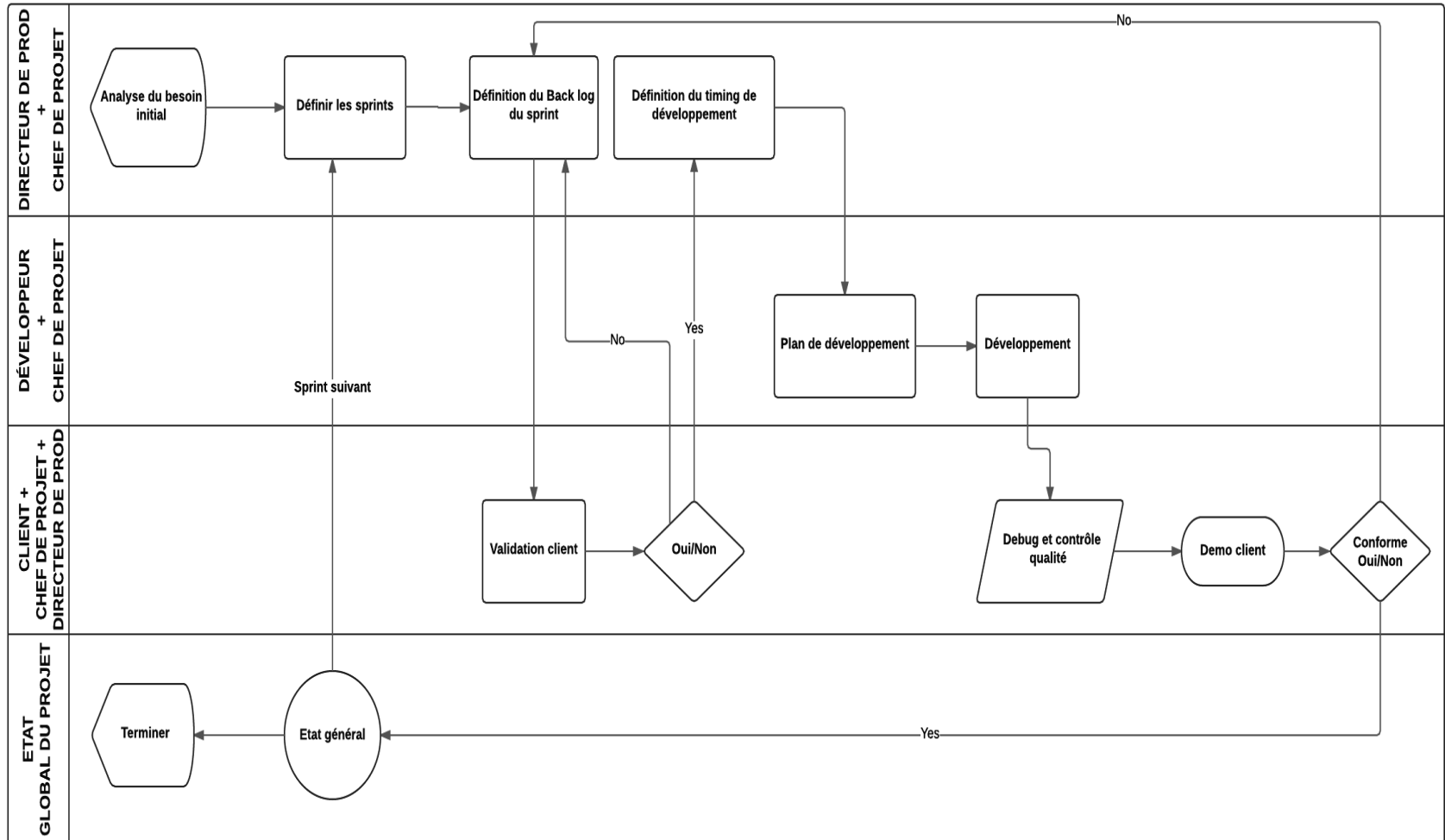
Stage d'observation de
Vanmoortel Nolan
2017

Scrum

Une illustration

PROCESSUS DE DÉVELOPPEMENT

Vanmoortel Nolan | ITDM sprl



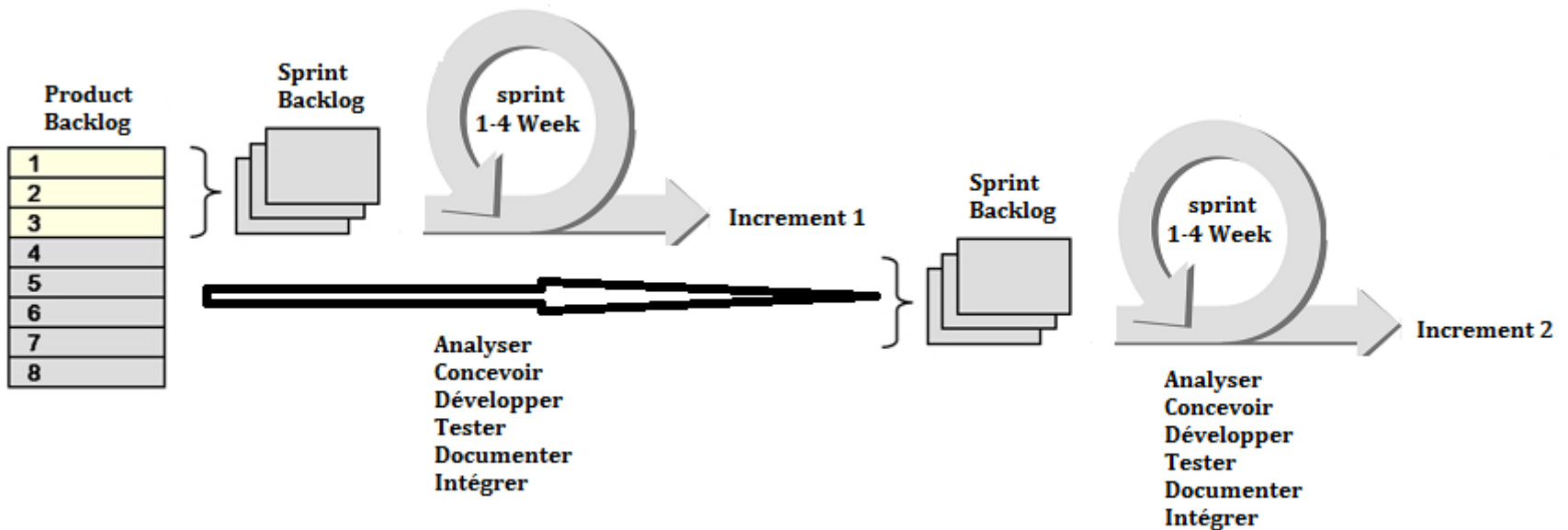
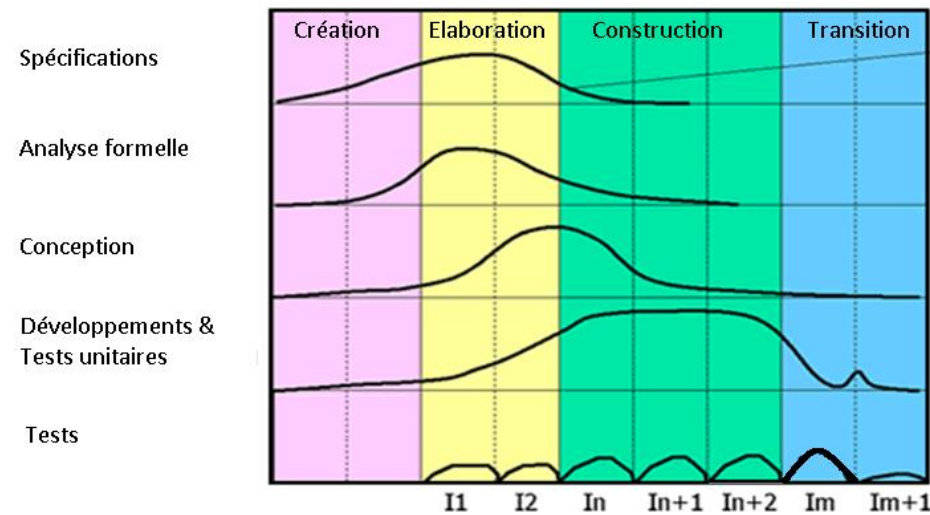
Livrable	Quand
1. Rapport d'analyse initiale	S3
2. Implémentation architecture : revue du code en séance	S6
3. Revue du code en séance	S9
4. Démo d'avancement en séance	S9
5. Livrable supprimé	S10
6. Code de tout le projet + Rapport + Démo	S12

Projet AE - PU ou Scrum ?

Comparaison PU et Scrum

PU

versus Scrum



PU

versus Scrum

- Définition complète des objectifs du projet
- 4 grandes phases
- Planning : date de fin du projet définie
- Outputs très bien définis

- Backlog
- Chaque itération couvre le cycle complet pour les US
- Product owner détermine quand le projet est fini
- Output : code fonctionnel et documenté, backlog à jour

Ce que vous allez faire seuls pour préparer l'examen.

- Comparer les méthodes (slides sur moodle) pour pouvoir choisir une méthode.

Ce que nous ne ferons pas cette année.

- Gestion de configuration.
- Gestion des changements.
- Gestion du risque.
- Chapitre sur la qualité.