

Approche Agile

Les méthodes agiles

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

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

Principes Agile (1)

- Plus haute priorité : satisfaire le **client**
 - Livrer rapidement des fonctionnalités à grande valeur ajoutée
 -  • Impliquer le client dans le développement
 -  • Collaboration continue des utilisateurs (ou de leurs représentants) et des développeurs.
- Livraisons **incrémentales** fréquentes d'un logiciel opérationnel avec des cycles de quelques semaines à quelques mois.

Principes Agile (2)

A

- Accueillir les **changements** de besoins, même tard dans le projet pour donner un avantage compétitif au client
 - Conception orientée évolution.
- Soutien et confiance à **l'équipe** → personnes motivées
 - Equipes auto-organisées, pas de processus imposé
 - Réflexion de l'équipe aux moyens de devenir plus efficace & comportement adaptable
- **Rythme de développement soutenable** : maintenir indéfiniment un rythme constant.

Principes Agile (3)

- Attention continue à l'excellence technique et à une bonne conception
- Simplicité - càd l'art de minimiser la quantité de travail inutile
- Un logiciel opérationnel : principale mesure d'avancement
- Importance des tests (!! sont parfois amenés à jouer le rôle de spécifications)

Agile : adaptatif

Deux aspects :

- **fonctionnellement**, par adaptation systématique du produit aux changements de besoin
- **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

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

eXtreme Programming

An Agile method

XP : une méthode Agile

- Premier projet en 1996
- Orienté vers la **satisfaction du client** :
 - Livraison correspondant au **besoin réel** du client
 - **Gestion des changements**, même très tard dans le développement du projet
- Encourage la **collaboration** : clients, managers et développeurs : collaborateurs
- **Auto-organisation** : par l'équipe pour résoudre les problèmes de manière efficace

Agile

eXtreme Programming

Valeurs

1. Communication

- Communication directe plutôt qu'échanges de documents formels
- Développeurs travaillent directement avec le client
- ➡ A • Programmation en binôme (revue de code permanente)

2. Simplicité :

- Conception simple = application facile à maintenir
- ➡ A • Anticiper les extensions futures est une perte de temps, on va peut-être développer une complexité inutile
- Quelle est la solution la plus simple qui peut fonctionner ?
- Le client devrait aussi d'adhérer à ce principe pour ne pas demander des choses complexes qui ne seront jamais utilisées !

XP Valeurs

3. Feedback

- Tests dès le premier jour, livraison très tôt dans le projet et retour d'information du client
- Maximum de feedback pour corriger la trajectoire au plus tôt

4. Respect

- Pour la contribution de chacun

5. Courage

- Apporter des changements dès que suggérés
- Changer l'architecture d'un projet
- Jeter du code pour en produire un meilleur
- Essayer une nouvelle technique
- Accepter et traiter de front les mauvaises nouvelles

Pratiques XP

- **Livraisons fréquentes**
 - Mise en production rapide d'une version minimale du logiciel
 - Ensuite nouvelles livraisons incrémentales
- **Planification itérative**
 - Plan de développement au début du projet
 - Revu et remanié à chaque nouvelle itération
- **Client sur site**
 - Avec l'équipe de développement !
- **Rythme durable**
 - Tout au long du projet (semaine de 38h !)

Pratiques de programmation

- Conception simple

- Refactoring



- Code en permanence remanié pour rester simple et compréhensible

- Tests unitaires

- Batteries de tests de non-régression qui permettent de faire face aux modifications permanentes

- Tests de vérification / d'acceptation

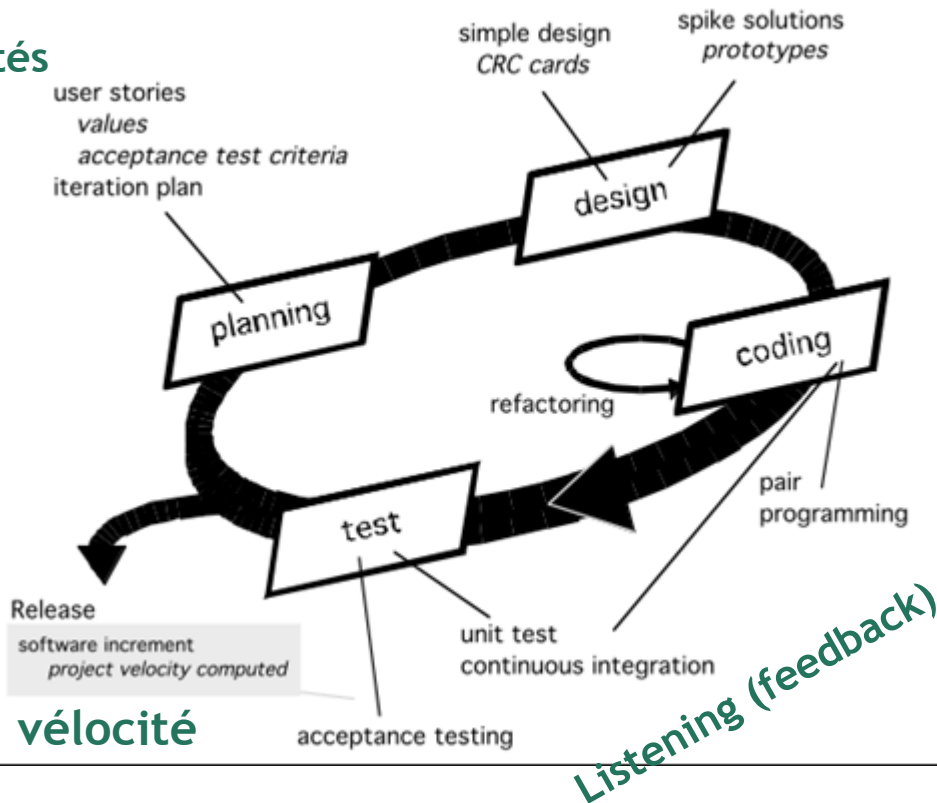
- Batteries de tests automatisées

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



Itération :
+/- 2 semaines

Calcul du
temps de
travail

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

User stories

- 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
- Format associé aux méthodes agiles
- Implémentée en une seule itération!

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

ID

Nom

User stories : si PAE Agile

Créer une entreprise

Histoire

- En tant que **utilisateur**

Je veux **créer une entreprise et une personne de contact**

Afin de **pouvoir l'inviter à la prochaine JE**

Validation

- Code postal en 4 chiffres
- Nom de la commune correspondant au code postal
- Une personne de contact a un numéro de tel et/ou un email

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

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**.

Questions ?

<http://www.scrumalliance.org>

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

<http://www.journaldunet.com/developpeur/tutoriel/theo/071011-methode-agile-scrum.shtml>

Scrum

An Agile framework

Scrum

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

Scrum est adaptatif :

Agile

- 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.

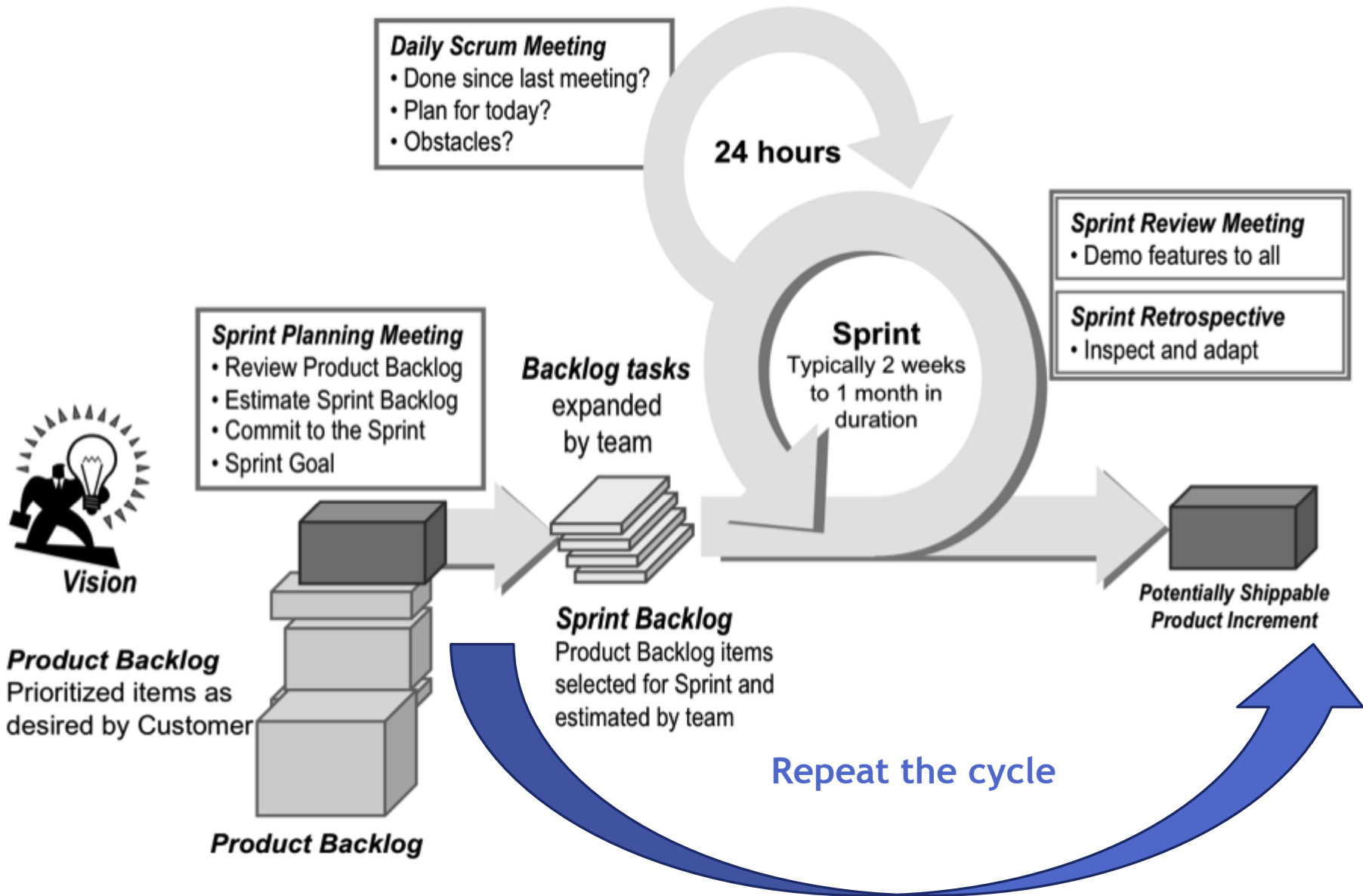
Rôles

- **Product Owner : directeur de produit,**
 - Représentant du client et des utilisateurs dans l'équipe
 - Expert dans le domaine
 - Ayant la vision du produit et l'autorité pour donner les priorités aux requirements du client
- **Scrum Master : gestionnaire**
 - Celui qui facilite l'application de Scrum dans l'équipe
 - Coach
- **Equipe :**
 - Performante, centrée sur les livraisons, auto-organisée
 - Ayant des compétences transversales
 - Rapportant les progrès

Concepts

- **Sprint** : unité de temps qui permet de rythmer les développements, correspondant à une **itération**
- **Product Backlog** : liste, *ordonnée par priorité*, des fonctionnalités pour le produit, définie par le directeur de produit
- **Sprint Backlog** : liste des fonctionnalités qui sont développées dans le sprint.

Cycle



Cycle

- 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 !

A l'intérieur d'un Sprint

- Développement d'un produit partiel :
 - Analyser
 - Concevoir
 - Développer
 - Tester
 - Documenter
 - Intégrer

Réunion de début de sprint

- **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

Réunion journalière

- **Scrum Meeting:**
 - Réunion une fois par jour à heure fixe
 - Mise en commun des apports de chacun
 - Partage des difficultés rencontrées
 - 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?

Réunion de revue

- Réunion de revue
 - Démonstration des nouvelles fonctionnalités
 - 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

Questions ?

Exercice : Le SVYpirale au cours de PAE : PU ou Scrum ?

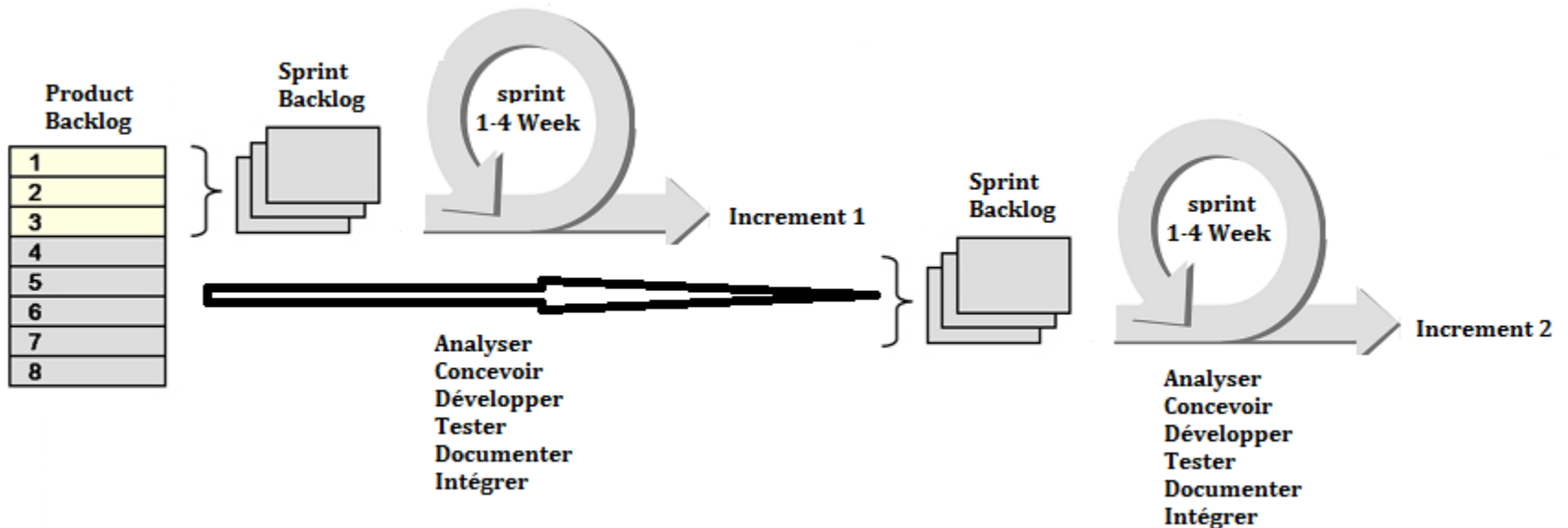
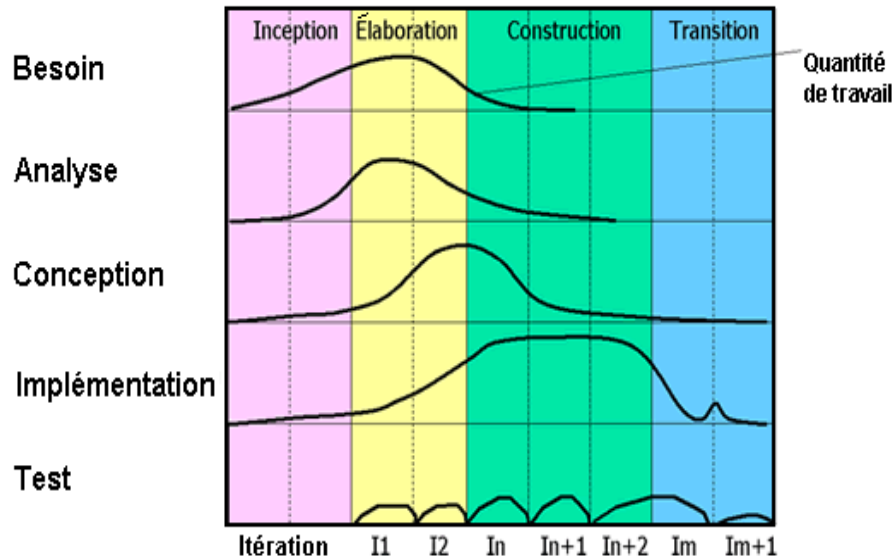
	t0	t1	t2	t2'	t2''	t3	t3'	t4	t5
		S3	S5		S7	S10		début S12	S12 S12
	Spécifications								
	Analyse formelle	Analyse formelle (uc) + corrections	Analyse formelle (C)			New Analyse formelle			
		Conception	Conception (C)			-			
		Codage (Connexion)	Codage	Codage	Codage		Codage		
		Vérification		Code review	Vérification		Vérification		
Output	Rapport d'analyse initiale	Implémentation architecture + rapport		Démonstration partielle	Implémentation du reste		Implémentation du changement	Rapport Final &	Démo

Validation

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 est un cycle complet
- Product owner détermine quand le projet est fini
- Output : code fonctionnel

L'ingénierie logicielle : choix d'une méthode

Cascade, V ou Y :

- Une approche très contrôlée du développement
 - Planification précise
 - Méthodes d'analyse et de conception
 - Documentation
 - Simple et facile à comprendre
 - Manque d'adaptabilité
 - Problèmes vus à la validation (client)
- Beaucoup de temps passés sur la façon de développer un système avant le développement lui même.

Incrémental & itératif:

Voir slides 3-2

- **Avantages**
 - Première version du système fournie rapidement
 - Risques d'échec diminués
 - Découverte des problèmes assez tôt
 - Parties importantes développées en premier et donc testées plus longuement
- **Inconvénients**
 - Difficulté de définir les itérations
 - Lourd à mettre en œuvre

Agile

- **Avantages**
 - Méthodes focalisées sur le développement
 - Méthodes basées sur une approche itérative
 - Livraison rapide & feed-back des clients
 - Développement d'applications dont les exigences changent
- **Inconvénients**
 - (Souvent) Aucune documentation
 - Difficulté à mettre en œuvre dans :
 - Affectation des priorités
 - Simplicité dans les changements additionnels
 - Implication intense des développeurs

Agile (2)

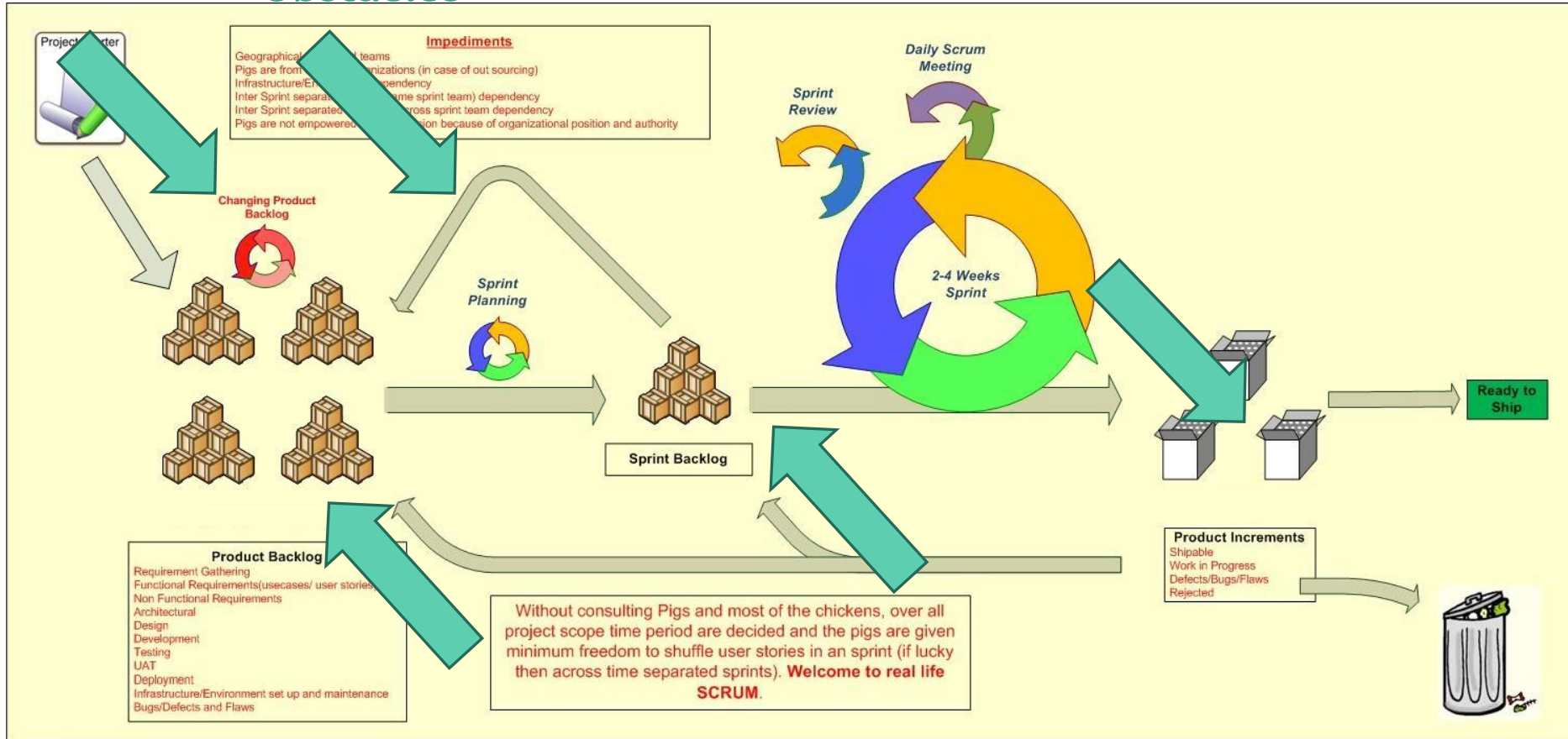
- **Frontière ténue** entre l'application de la méthode et le « n'importe quoi » qui peut advenir quand on travaille sans méthode

Inconvénients potentiels :

- Pas de documents d'analyse ou de spécifications détaillées
- Les tests d'acceptation remplacent les spécifications
- Emergence de l'architecture au fur et à mesure du développement
- Manque de contrôle et de structuration, risques de dérive

Scrum contesté :

Obstacles



<http://architecture-soa-bpm-eai.blogspot.be/2011/01/scrum-beautiful-and-ugly.html>

Lequel choisir ?

- Il n'existe pas un modèle idéal, un seul modèle à appliquer en toute situation.
- Un **compromis entre agilité et discipline** semble idéal.
- Adopter un cycle de vie est déjà une preuve de maturité pour l'entreprise !

Comparatif

	Discipline	Agile
Taille (de l'équipe)	<ul style="list-style-type: none"> Nécessaire si grosses équipes Difficile à adapter aux petits projets 	<ul style="list-style-type: none"> Petites équipes Forte dépendance au mode de connaissance tacite qui limite la taille DANGER
Criticité (pertes en cas de défaut)	<ul style="list-style-type: none"> Idéal pour gérer le développement de produits critiques Convient moins aux produits peu critiques 	<ul style="list-style-type: none"> Non testé sur des systèmes critiques Difficultés potentielles dues au simple design et au manque de documentation
Dynamisme (% de changement d'exigences/mois)	<ul style="list-style-type: none"> Plans détaillés et design précis sont bien adaptés aux environnements stables Mais source de travail importante quand l'environnement est changeant 	<ul style="list-style-type: none"> Simple design & refactoring continu sont bien adaptés aux environnements forts changeants Mais peut être source de travail supplémentaire dans un environnement stable
Personnel (expérience)	<ul style="list-style-type: none"> Demande du personnel expérimenté durant la définition du projet Utilisation de juniors par la suite 	<ul style="list-style-type: none"> Demande la présence d'une masse critique de personne expérimentées Risqué avec des juniors
Culture (chaos vs ordre)	<ul style="list-style-type: none"> Organisations où les rôles sont clairement définis dans des procédures reconnues Culture d'"ordre" 	<ul style="list-style-type: none"> Adapté aux organisations laissant une grande marge de liberté aux employés Culture de "chaos"

Exercices :

- 1. examen juin 2011**
- 2. stage d'observation 2014**

Une entreprise a choisi d'organiser le cycle de vie du développement de ses logiciels selon un diagramme en V et selon un processus itératif.

Cela signifie que la société prend dans un ensemble de requirements (demandes clients ou marché), ceux qui seront pris en compte pour une version développée à 6 mois.

Elle les fait traduire en spécifications, sur base desquelles démarrent les analyses, la conception technique, le codage et les tests unitaires.

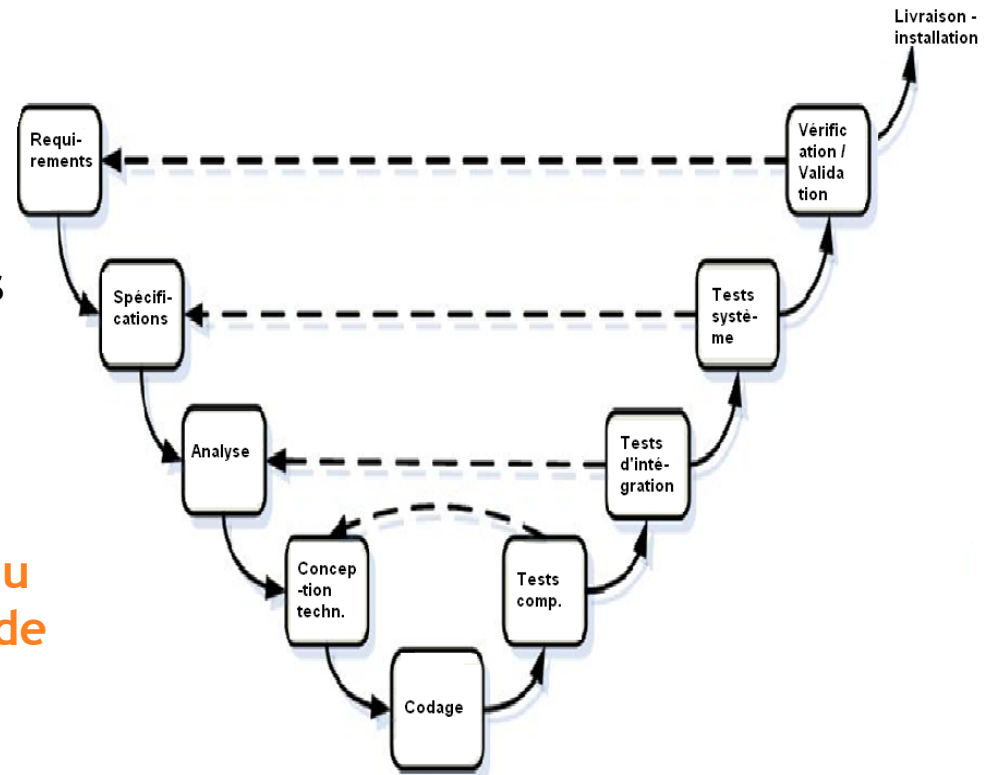
A chaque livraison intermédiaire (en moyenne toutes les deux semaines), les tests sont exécutés par une équipe de tests.

Donnez les avantages et inconvénients de cette pratique.

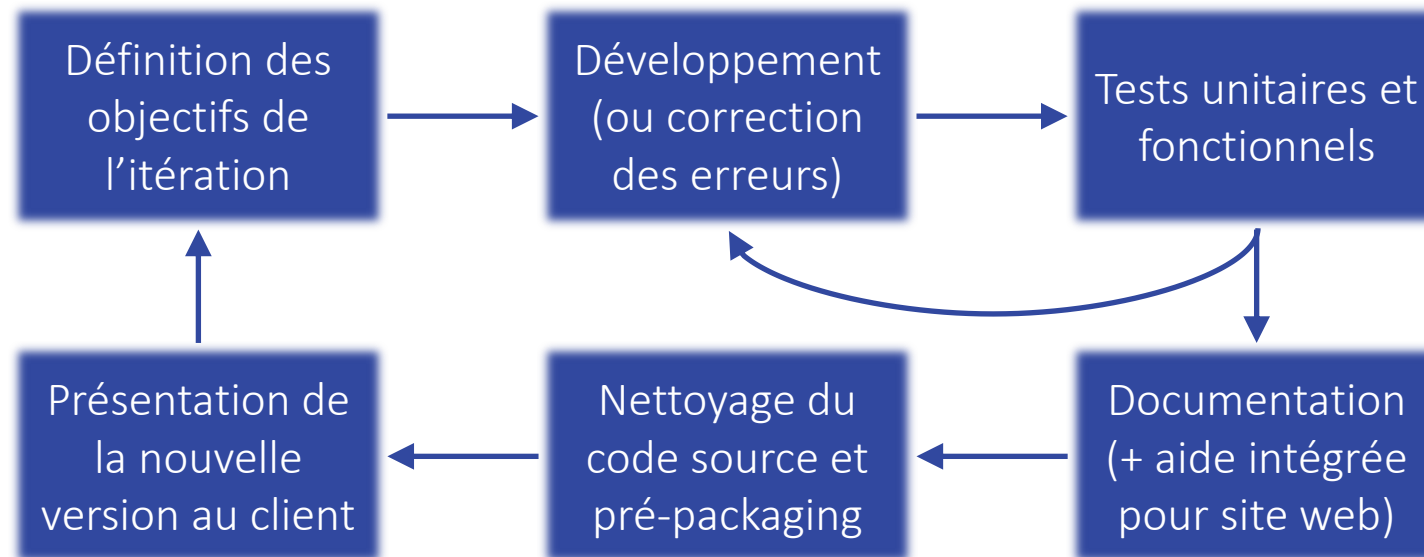
Maximum 25 lignes

Représentez ce cycle sur une ligne du temps où apparaissent des périodes de temps de 2 semaines et 6 mois.

ORGA entreprises



Identifiez ce cycle de vie. Justifiez.
Donnez les avantages et inconvénients de
cette pratique.



Questions - réponses