Les Méthodes Agiles

Valentin Lachand & Alix Ducros

Master CCI
Université Claude Bernard Lyon 1

Basé sur les cours d'Aurélien Tabard et de Yannick Prié

Plan

1. Retour rapide sur les méthodes de conception
2. Principes des méthodes Agiles
3. XP : eXtreme Programming
4. Scrum
5. Autres méthodes

Objectifs du cours

- Connaître différentes méthodes itératives
- Développer une pratique de travail réflexive
- Être capable d’appliquer SCRUM
- Être conscient des limites de ces méthodes

Lecture

- The Agile Manifesto
- Getting Real par 37 signals (plutôt sur le Lean)
- Scrum - Le guide pratique de la méthode agile la plus populaire

Plan

1. **Retour rapide sur les méthodes de conception**
2. Principes des méthodes Agile
3. XP : eXtreme Programming
4. Scrum
5. Autres méthodes

1. Retour rapide sur les méthodes de conception

Quelles méthodes connaissez vous?

1. Retour rapide sur les méthodes de conception

. Rien

. Cascade

. Modèle en V

. Itératif (RUP)

. Agile

Rien a.k.a. “code and fix”

Bénéfices :

. Marche si vous travaillez seul

. Pas de surcoûts dûs à la gestion de projet

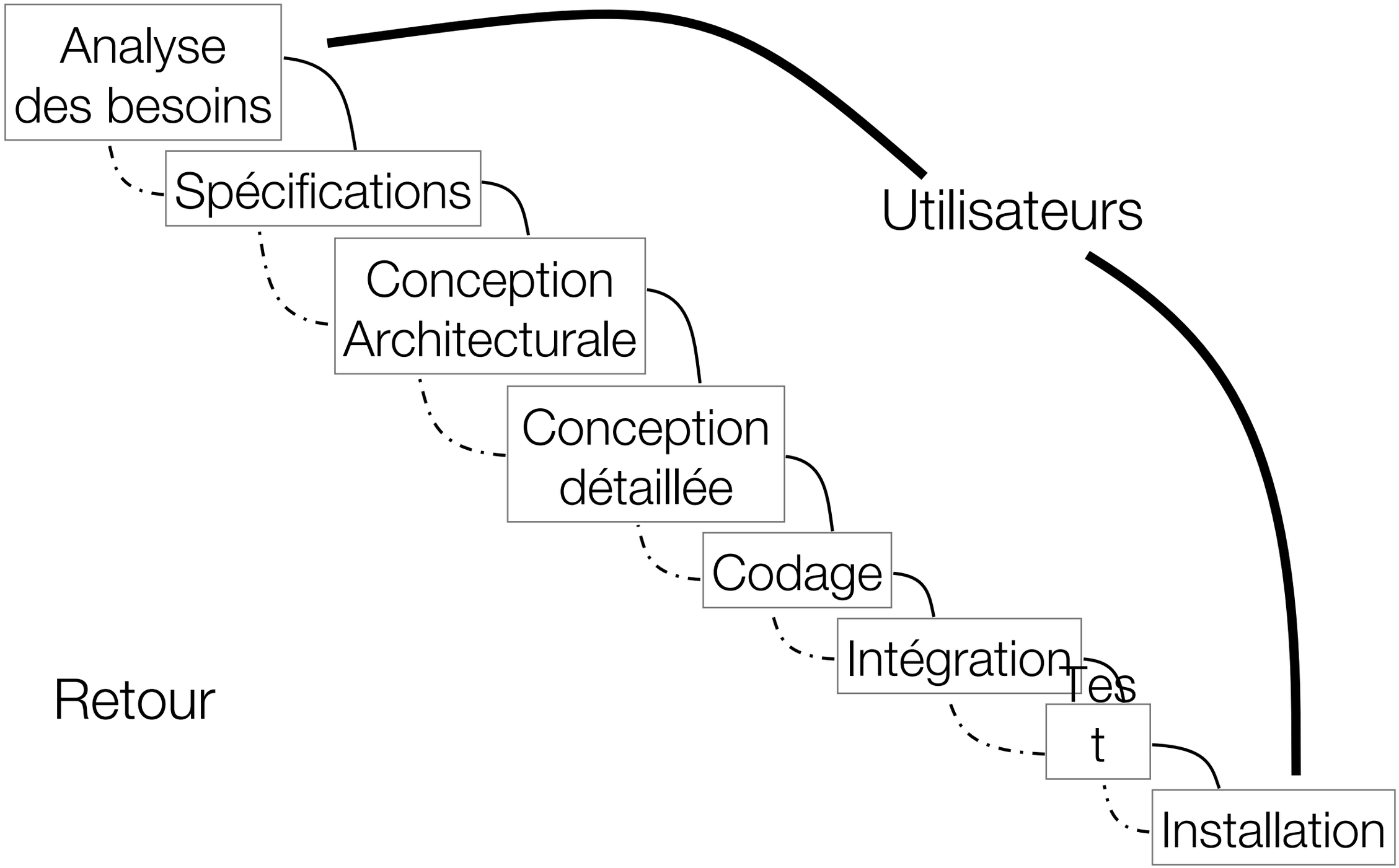
Inconvénients :

. Rapidement compliqué

. Pas de visibilité

. ...

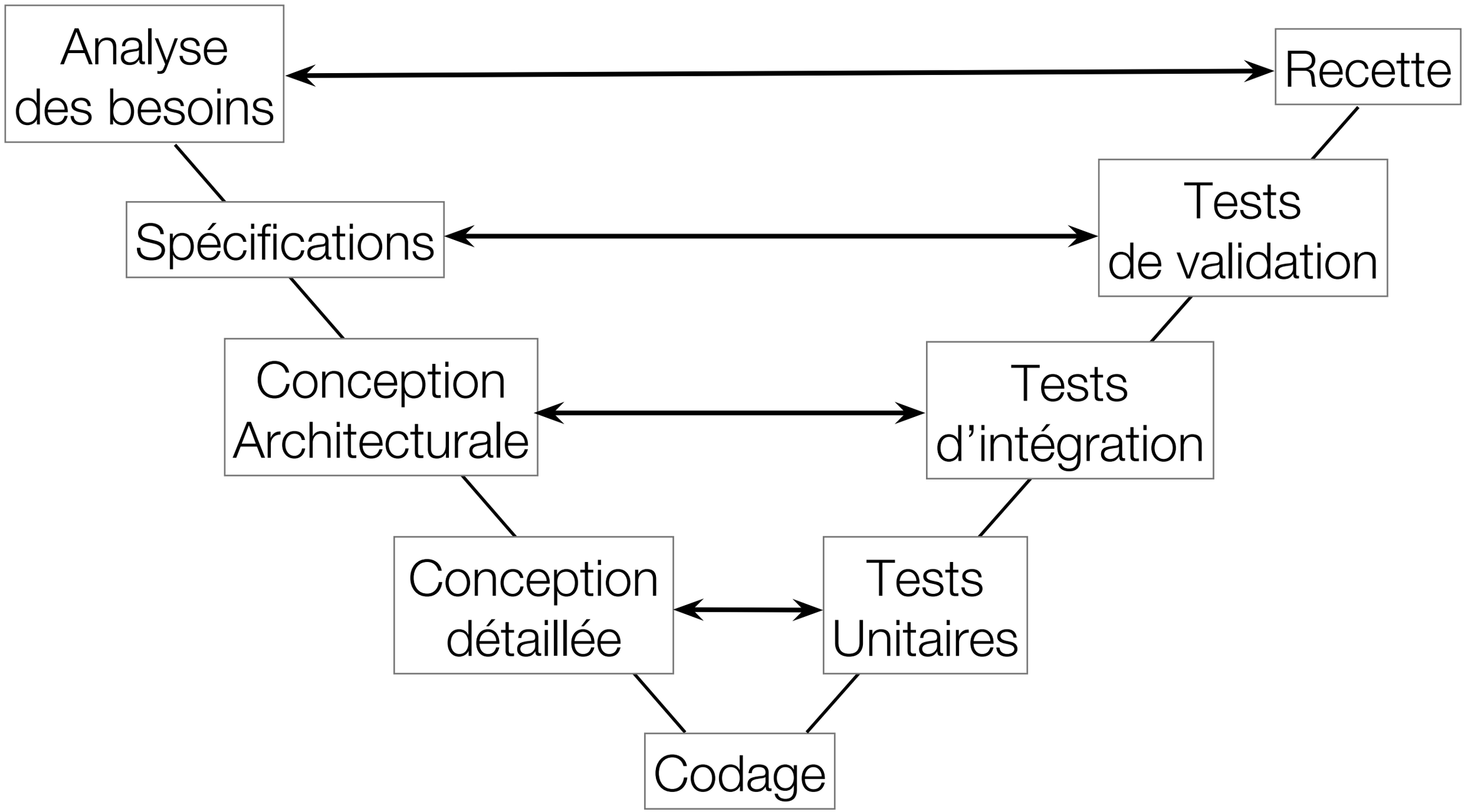
Cascade


Analyse des besoins
Spécifications
Conception Architecturale
Conception détaillée
Codage
Intégration
Test
Installation
Utilisateurs
Retour

Cascade

- Contre-exemple classique, existe peu en réalité
- Bénéfices :
  - tâches établies au début du projet
  - segmentation claire des responsabilités
  - passation de tâches aisées
- Inconvénients :
  - planification trop rigide alors que les tâches sont changeantes
  - la responsabilité du succès du projet n'est pas partagée
    si le projet échoue : "c'est pas de ma faute, j'ai bien fait ma part".
  - si une tâche n'est pas bien effectuée tout l'édifice est ébranlé
  - trop de choses sont faites qui ne sont pas directement liées au produit logiciel à construire

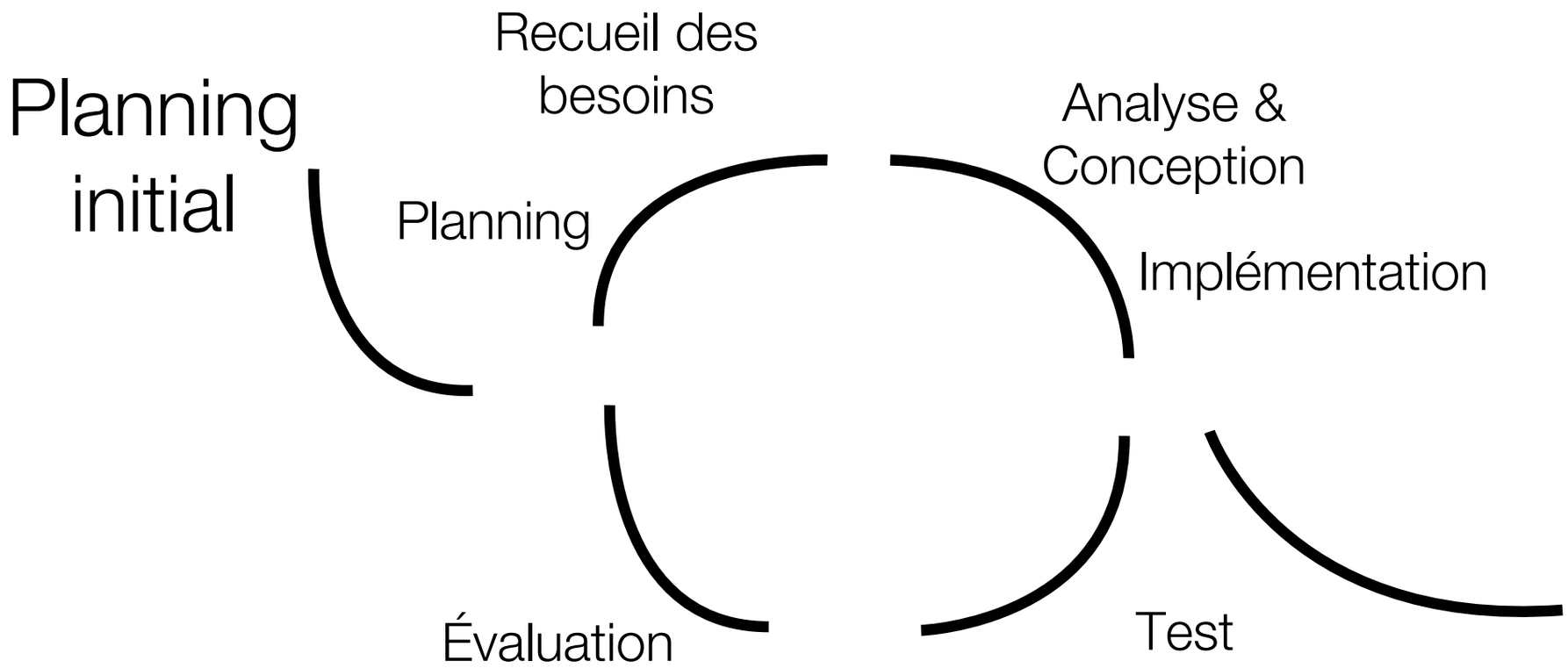
Modèle en V


Analyse des besoins
Recette
Spécifications
Tests de validation
Conception Architecturale
Tests d'intégration
Conception détaillée
Tests Unitaires
Codage

Modèle en V

- Bénéfices :
  - tâches clairement établies au début du projet
  - responsabilisation via des retours en cas d’erreur
- Inconvénients :
  - les tâches sont changeantes
  - la responsabilité du succès du projet n’est pas partagée

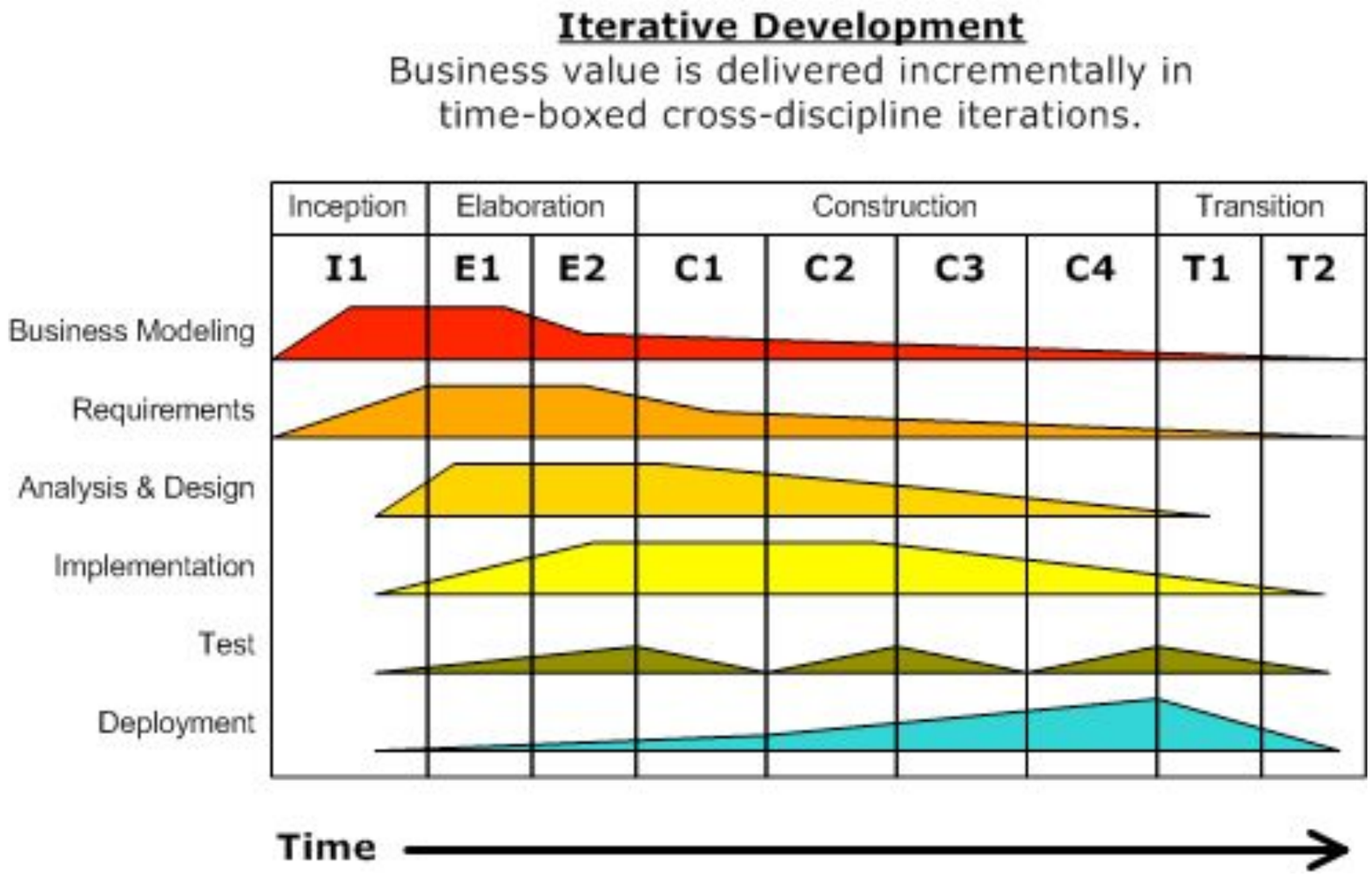
Itératif


Planning initial
Recueil des besoins
Analyse & Conception
Implémentation
Test
Évaluation
Planning

Itératif : le cas de Rational Unified Process

- Itératif et incrémental
- Dirigé par les Cas d’Utilisations
- Centré sur l’architecture
- Focus sur le risque

Itératif : le cas de Rational Unified Process



Iterative development illustration par Dutchguilder
https://en.wikipedia.org/wiki/File:Development-iterative.gif

Rational Unified Process : principes

. Développer de manière itérative,
avec le risque dans les 1^{e} itérations
. Gérer les besoins
. Utiliser une architecture basées sur les composants
. Modéliser le logiciel visuellement
. Vérifier la qualité en continu (test)
. Contrôler les changements

Méthodes agiles

Trouver un compromis :

- le minimum de méthode permettant de mener à bien les projets en restant agile
- capacité de réponse rapide et souple au changement
- orientation vers le code plutôt que la documentation

Pourquoi être agile

Exemples concrets :

. Cas innovation, grand fabricant de stylo

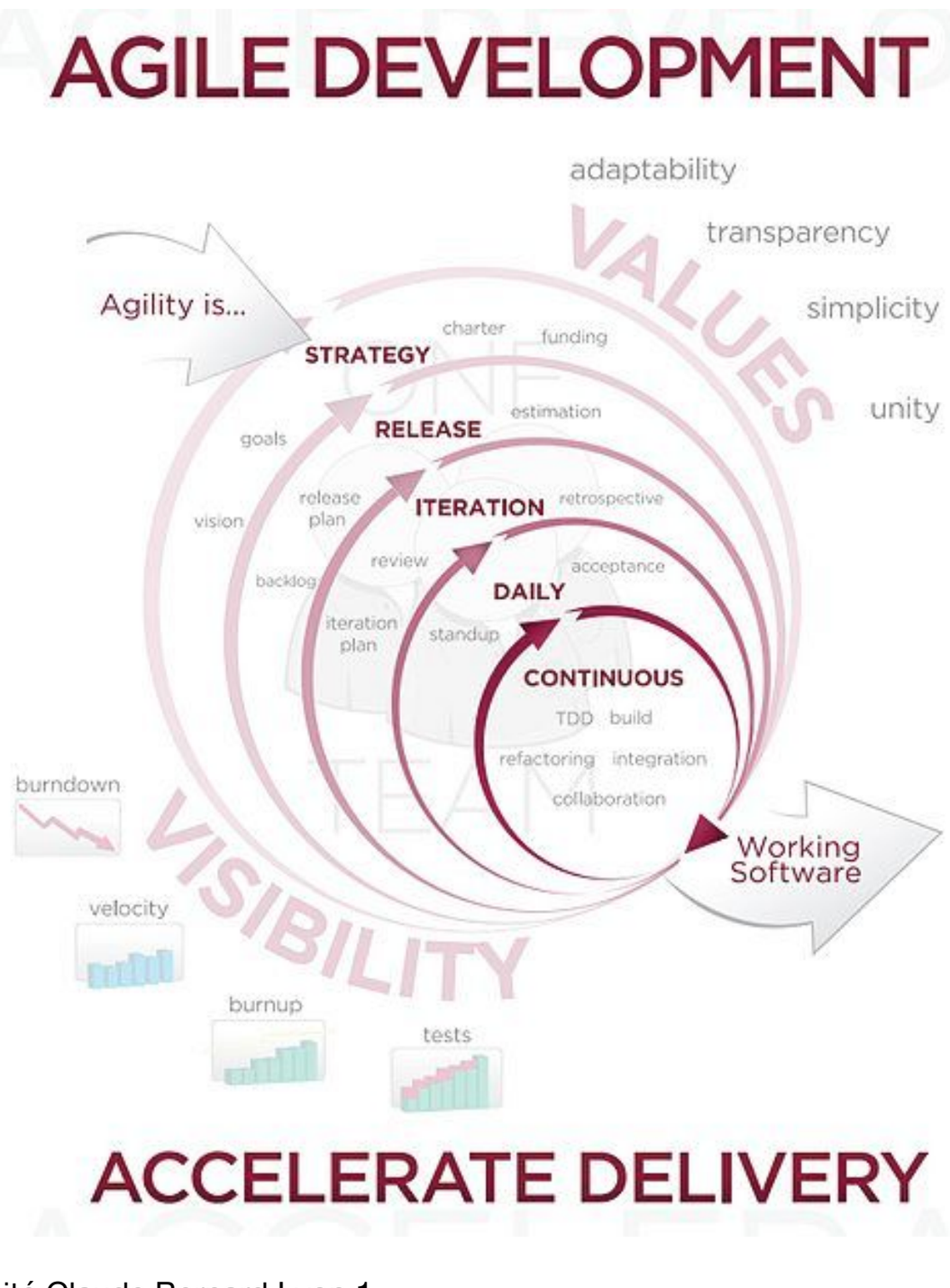
. Cas service, embauché pour un site web

. Cas client avec un problème mais sans solution

. Cas domaine complexe : les transports

Agile

Attribution: VersionOne, Inc.
http://commons.wikimedia.org/wiki/File:Agile_Software_Development_methodology.jpg



Historique

- Années 90
  - réaction contre les grosses méthodes
  - prise en compte de facteurs liés au développement logiciel
- Fin années 90: développement de méthodes
  - d’abord des pratiques liées à des consultants, puis des livres
  - XP, Scrum, FDD, Crystal…
- 2001
  - les principaux méthodologues s’accordent sur le « Agile manifesto »
- Depuis
  - projets Agile mixent des éléments des principales méthodes

Plan

1. Retour rapide sur les méthodes de conception
2. **Principes des méthodes Agile**
3. XP : eXtreme Programming
4. Scrum
5. Autres méthodes

2. Principes des méthodes Agile

. Les principes communs

. Le Manifeste

. Agile et modélisation

Principes communs aux méthodes agiles

- Méthodes adaptatives (vs. prédictives)
  - itérations courtes
  - lien fort avec le client
  - fixer les délais et les coûts, mais pas la portée
- Insistance sur l’humain
  - les développeurs sont des spécialistes, et pas des unités interchangeables
  - attention à la communication humaine
  - équipes auto-organisées
- Processus auto-adaptatif
  - révision du processus à chaque itération

Méthodes

- Simplicité
- Légèreté
- Orientées participants plutôt que plan
- Nombreuses :
  - eXtreme Programming
  - SCRUM
  - ...
- Pas de définition unique mais un manifeste

Le manifeste Agile (The Agile Manifesto)

- Février 2001, rencontre et accord sur un manifeste
  - Mise en place de la « Agile alliance »
  - Objectif : promouvoir les principes et méthodes Agile
  - http://www.agilealliance.com/
- Les signataires privilégient
  - les individus et les interactions davantage que les processus et les outils
  - les logiciels fonctionnels davantage que l'exhaustivité et la documentation
  - la collaboration avec le client davantage que la négociation de contrat
  - la réponse au changement davantage que l'application d'un plan
- 12 principes

Manifeste Agile : principes

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile process harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Manifeste Agile : principes

7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design enhances agility
10. Simplicity – the art of maximizing the amount of work not done – is essential
11. The best architectures, requirements, and designs emerge from self-organizing teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Processus agile et modélisation (Larman 2005, p.36-37)

- Utilisation d'UML
- La modélisation vise avant tout à **comprendre** et à **communiquer**
- Modéliser pour les parties inhabituelles, difficiles ou délicates de la conception.
- Rester à un niveau de modélisation minimalement suffisant
- Modélisation en groupe
- Outils simples et adaptés aux groupes
- Les développeurs créent les modèles de conception qu'ils développeront

Plan

1. Retour rapide sur les méthodes de conception
2. Principes des méthodes Agile
3. **XP : eXtreme Programming**
4. Scrum
5. Autres méthodes

Historique de l'eXtreme Programming

- 1996 : Ward Cunningham et Kent Beck
- Projet Chrysler Comprehensive Compensation (C3)
- Réorganisation du système de paie
- Propagation mondiale grâce à Internet
- Implantation lente en France

Principes de l'XP

http://www.extremeprogramming.org/

Dimension humaine considérée comme déterminante pour la réussite de tout projet.

Et :

- Feedback rapide et constant
- Compréhension partagée
- Bien-être de l'équipe
- Processus fluide et continu

Caractéristiques principales

. Le client (maîtrise d'ouvrage) pilote lui-même le projet, et ce de très près grâce à des cycles itératifs extrêmement courts (1 ou 2 semaines).

. L'équipe autour du projet livre très tôt dans le projet une première version du logiciel, et les livraisons de nouvelles versions s'enchaînent ensuite à un rythme soutenu pour obtenir un feedback maximal sur l'avancement des développements.

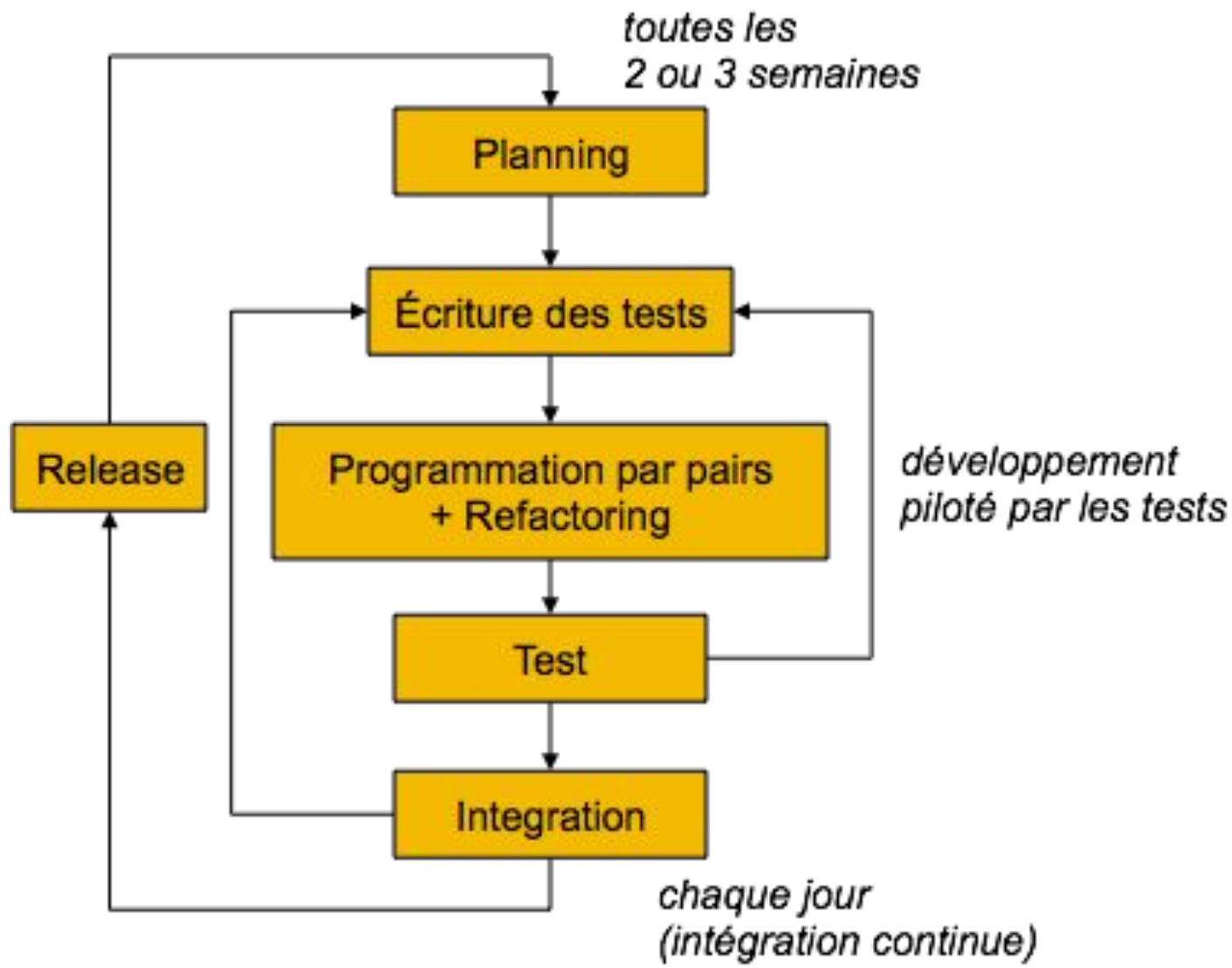
. L'équipe s'organise elle-même pour atteindre ses objectifs, en favorisant une collaboration maximale entre ses membres.

. L'équipe met en place des tests automatiques pour toutes les fonctionnalités qu'elle développe, ce qui garantit au produit un niveau de robustesse très élevé.

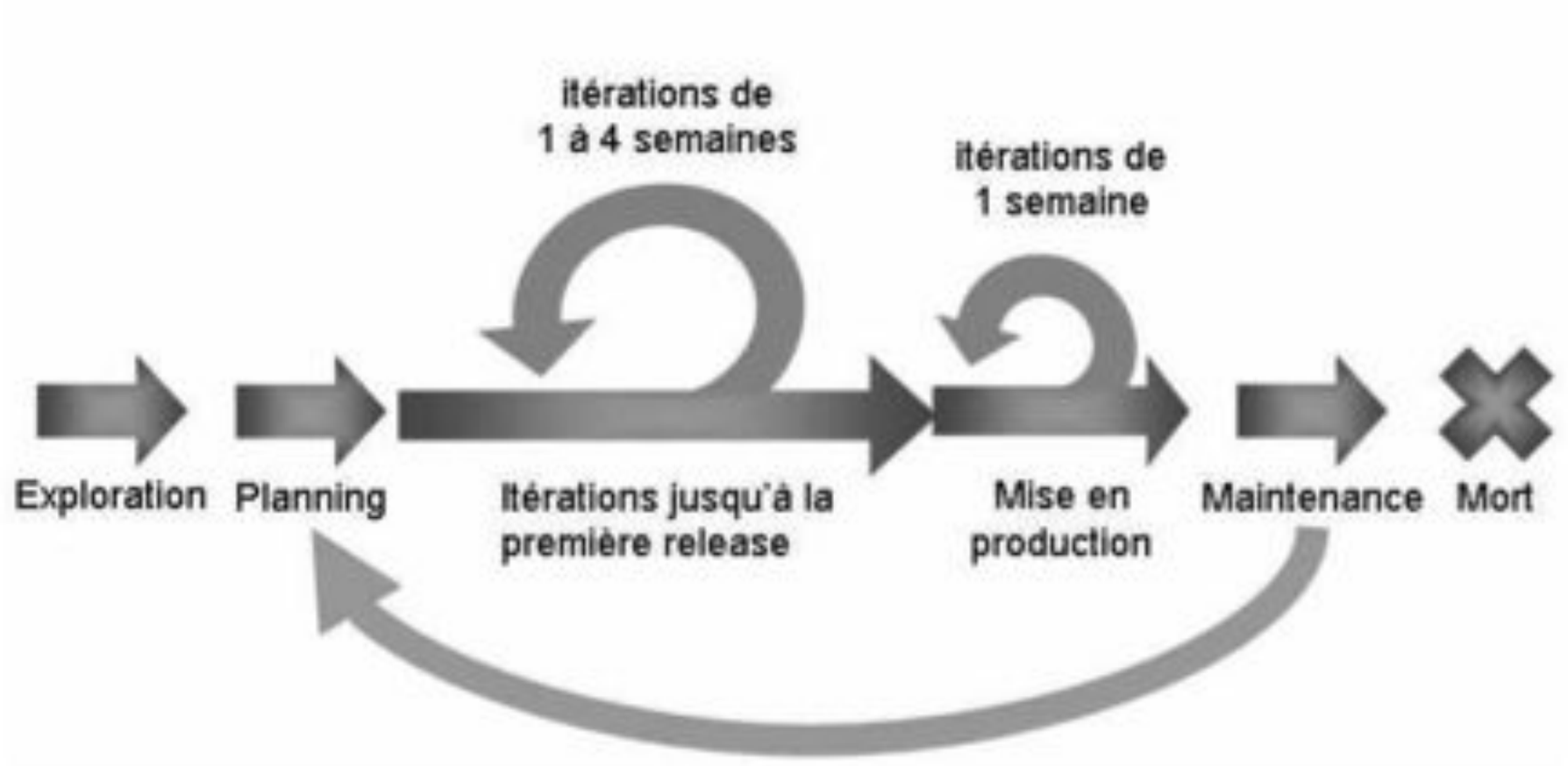
. Les développeurs améliorent sans cesse la structure interne du logiciel pour que les évolutions y restent faciles et rapides.

(extrait de http://www.design-up.com/methodes/XP/)

Vue d’ensemble



Planification globale d'un projet



Comité de pilotage

- Comité de pilotage
  - dirige les aspects stratégiques du projet,
  - définit les objectifs
  - et alloue les moyens nécessaires
- Client interlocuteur :
  - le plus à même de connaître le besoin et de l’exprimer
  - « Client interlocuteur » = appui du « sponsor principal »
    - directeur des systèmes d’information
    - ou porte-parole officiel de l’entreprise disposant d’une autorité suffisante pour légitimer le projet et incarner l’engagement de l’entreprise
- En phase de préparation, il faut s’assurer de:
  - La clarté des objectifs du projet
  - Leur compréhension par toutes les parties prenantes
  - Accord de l’entreprise cliente

Bien-être de l’équipe

- Pratique humaniste favorisant
  - l'élimination progressive des risques
  - l’établissement d’un rythme de travail régulier, sans heures supplémentaires
- But
  - éliminer le stress,
  - éliminer les tensions a l’intérieur du groupe,
  - éliminer les tensions entre le groupe et les donneurs d’ordres,
  - éliminer le risque de défection.

Structure d’une équipe XP

Les profils :

- Client
- Testeur
- Manager
- Coach
- Tracker
- Programmeur

Le rôle de chaque profil (1/3)

- Le client :
  - Membre à part entière de l’équipe
  - Présence physique imposé dans l’équipe tout au long du développement
  - Spécifie les fonctionnalités à implémenter et tests fonctionnels
  - Rôle pouvant être tenu par une ou plusieurs personnes
- Le testeur :
  - Assistant du client
  - Un programmeur
  - Implémente (code) les tests de recette le plus tôt possible, valide le fonctionnement du code et vérifie la non régression

Le rôle de chaque profil (2/3)

- Le manager :
  - Responsable de l'infrastructure dans laquelle l'équipe travaille
  - S'assure la non existence de problèmes étrangers au projet ou logistiques (espace de travail, outillage, documentation, etc.)
- Le coach :
  - S'assure de la bonne compréhension de la méthode XP et de son application correcte par les différents acteurs du projet
  - Généralement « expert méthode » et bon communicateur doublé d'un technicien crédible et respecté
  - A pour objectif de se faire oublier

Le rôle de chaque profil (3/3)

- Le tracker :
  - Contrôle l'avancement des tâches à l'intérieur d'une itération.
  - S'entretient fréquemment avec chaque programmeur pour s'enquérir des difficultés rencontrées et du travail déjà effectué.
  - Construit régulièrement une vision fiable de l'avancement des tâches afin de détecter le plus tôt possible les dérives et de lancer une nouvelle phase si nécessaire
- Le programmeur :
  - Estime la charge nécessaire à l'implémentation d'un scénario dans le cadre du jeu de la planification.
  - Implémente les scénarios en s'appuyant sur l'écriture de tests unitaires
  - Est aussi analyste, concepteur.

XP : Pratiques (1)

- Le « jeu de la planification »
  - regroupement des intervenants pour planifier l'itération
  - les développeurs évaluent les risques techniques et les efforts prévisibles liés à chaque fonctionnalité (user story, sortes de scénarios abrégés)
  - les clients estiment la valeur (l'urgence) des fonctionnalités, et décident du contenu de la prochaine itération
- Temps court entre les releases
  - au début : le plus petit ensemble de fonctionnalités utiles
  - puis : sorties régulières de prototypes avec fonctionnalités ajoutées
- Métaphore
  - chaque projet a une métaphore pour son organisation, qui fournit des conventions faciles à retenir

XP : Pratiques (2)

- Conception simple
  - toujours utiliser la conception la plus simple qui fait ce qu'on veut
    - doit passer les tests
    - assez claire pour décrire les intentions du programmeur
  - pas de généricité spéculative
- Tests
  - développement piloté par les tests : on écrit d'abord les tests, puis on implémente les fonctionnalités
  - les programmeurs s'occupent des tests unitaires
  - les clients s'occupent des tests d'acceptation (fonctionnels)
- Refactoring
  - réécriture, restructuration et simplification permanente du code
  - le code doit toujours être propre

Exemple rapide : Test Driven Development

```
public int add(a,b) {
      return 0;
}
```

```
@Test
public final void testAdd() {
      int a, b, res;
      a = 5;
  b  = 5;
  res = a + b;
      assertTrue("a et b positif", add(a, b) == res);
  }
```

Exemple : Les tests

```
public int add(a,b) {
        return a+b;
}
```

```
@Test
public final void testAdd() {
        int a, b, res;
        a = 5;
   b  = 5;
   res = a + b;
        assertTrue("a et b positif", add(a, b) == res);
   }
```

XP : Pratiques (3)

. Programmation par paires (pair programming)
  . tout le code de production est écrit par deux programmeurs devant un ordinateur
  . l'un pense à l'implémentation de la méthode courante, l'autre à tout le système
  . les paires échangent les rôles, les participants des paires changent
  . permet de
  . se contrôler mutuellement
  . diminuer les erreurs de conception
  . mieux se concentrer sur le travail
  . éliminer le risque de dépendance à un développeur
  . lisser les inégalités d'expériences en associant « confirmé & débutant »
  . faire circuler la connaissance à l'intérieur du projet

XP : Pratiques (4)

- Propriété collective du code
  - tout programmeur qui voit une opportunité d'améliorer toute portion de code doit le faire, à n'importe quel moment
- Intégration continue
  - utilisation d'un gestionnaire de versions (e.g., CVS)
  - tous les changements sont intégrés dans le code de base au minimum chaque jour : une construction complète (build) minimum par jour
  - 100% des tests doivent passer avant et après l'intégration

XP : Pratiques (5)

- Semaine de 40 heures (35 en France)
  - les programmeurs rentrent à la maison à l'heure
  - faire des heures supplémentaire est signe de problème
  - moins d'erreurs de fatigue, meilleure motivation
- Des clients sur place
  - l'équipe de développement a un accès permanent à un vrai client/utilisateur (dans la pièce d'à côté)
- Des standards de codage
  - tout le monde code de la même manière
  - tout le monde suit les règles qui ont été définies
    - il ne devrait pas être possible de savoir qui a écrit quoi

XP : Pratiques (6)

. Règles
    . l’équipe décide des règles qu’elle suit, et peut les changer à tout moment
. Espace de travail
    . tout le monde dans la même pièce
        . awareness
    . tableaux au murs
    . matérialisation de la progression du projet
        . par les histoires (user stories) réalisées et à faire
            . papiers qui changent de position, sont réorganisés
        . par les résultats des tests
        . ...

Unification des savoirs (1/2)

- Formation
- Mise à niveau des ressources avant le lancement du projet :
  - Phase d’apprentissage théorique par un formateur professionnel
  - Mise en pratique dans le cadre d’un projet pilote
- Projet partagé en 2 phases :
  - Phase coaching, peu productive, avec pour objectifs de
    - réaliser en situation le potentiel de chaque membre
    - gérer les parcours individualisé avec le soutien pédagogique de moniteurs (les coachs)
  - Phase productive lors de la première phase du vrai projet

Unification des savoirs (2/2)

. Montée en compétence de l’équipe

. Objectifs :

  . Maximiser la rentabilité de l’investissement pour la montée en compétences des ressources

  . Éviter l’écueil 80/20 : l’acquisition des 20% de connaissances les plus pointues requiert 80% des efforts (et de l’investissement) en formation.

  . Faire confiance au temps et considérer l’uniformisation des compétences collectives

XP : quelques point du site web

Planning

- User stories are written.
- Release planning creates the schedule.
- Make frequent small releases.
- The Project Velocity is measured.
- The project is divided into iterations.
- Iteration planning starts each iteration.
- Move people around.
- A stand-up meeting starts each day.
- Fix XP when it breaks.

Designing

- Simplicity.
- Choose a system metaphor.
- Use CRC cards for design sessions.
- Create spike solutions to reduce risk.
- No functionality is added early.
- Refactor whenever and wherever possible.

Coding

- The customer is always available.
- Code must be written to agreed standards.
- Code the unit test first.

XP : avantages

- Concept intégré et simples
- Pas trop de management
  - pas de procédures complexes
  - pas de documentation à maintenir
  - communication directe
- Programmation par paires
- Gestion continuelle du risque
- Estimation permanente des efforts à fournir
- Insistance sur les tests : facilite l’évolution et la maintenance

XP : inconvénients

Approprié pour de petites équipes (pas plus de 10 développeurs), ne passe pas à l'échelle

- pour des groupes plus gros, il faut plus de structure et de documentation (ceremony)

Risque d'avoir un code pas assez documenté

- des programmeur qui n'auraient pas fait partie de l'équipe de développement auront sans doute du mal à reprendre le code

Pas de design générique

- pas d'anticipation des développements futurs

Plan

1. Retour rapide sur les méthodes de conception
2. Principes des méthodes Agile
3. XP : eXtreme Programming
4. **Scrum**
5. Autres méthodes

Scrum

- Scrum : la mêlée (au rugby)
- Part de l’observation que le client change d’avis et de besoins en cours de projet
- Prend en compte l’aspect incertain de la gestion de projet
- Approche empirique

Phases de Scrum

1. Initiation / démarrage
    . Planning
        . définir le système : product Backlog = liste
        de fonctionnalités, ordonnées par ordre de priorité et d'effort
    . Architecture
        . conception de haut-niveau
2. Développement
    . Cycles itératifs (sprints) : 30j
        . amélioration du prototype
3. Clôture
    . Gestion de la fin du projet : livraison…

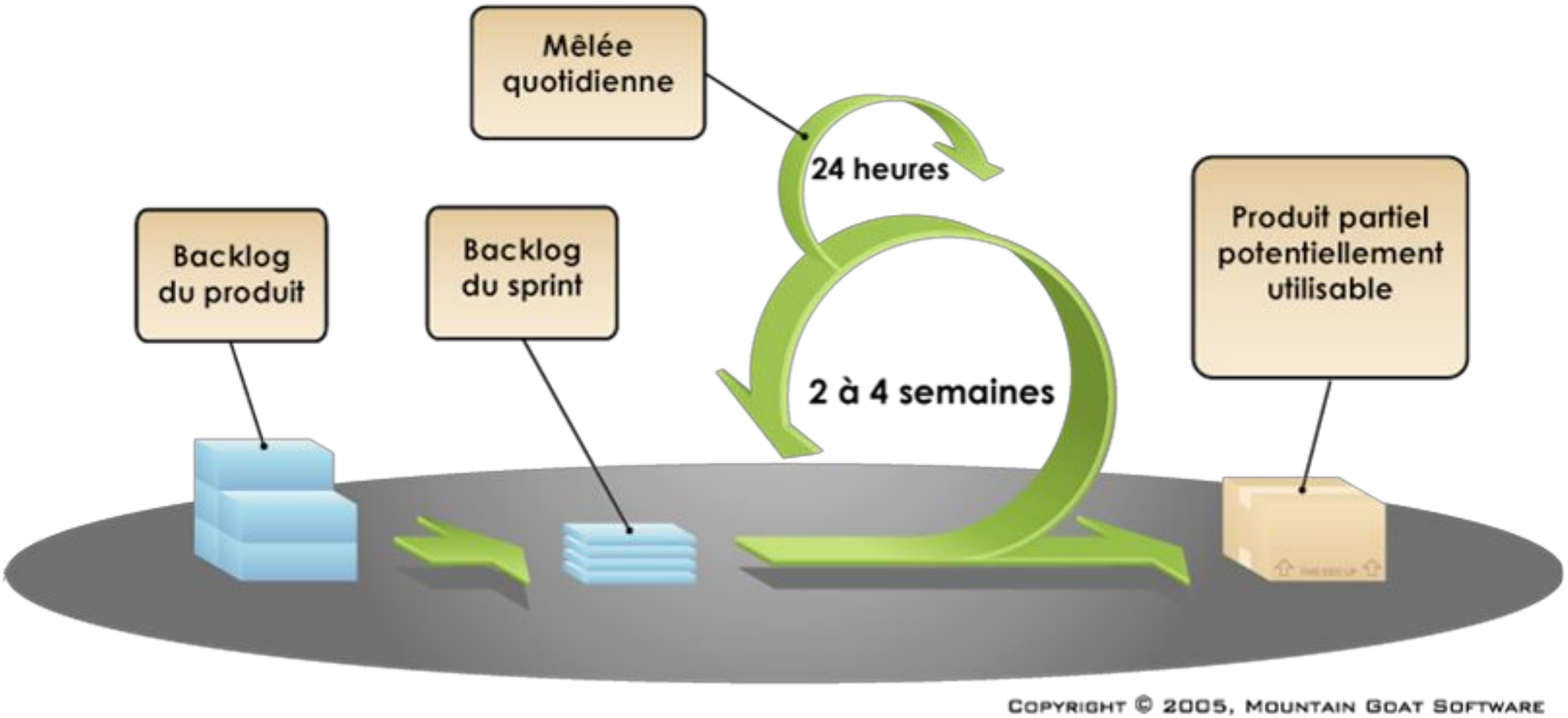
Exemple d’outil de planning : Planning Poker

- Utilisé pour estimer la **difficulté de la tâche**
- Chacun vote avec une carte
- Les valeurs extrêmes expliquent leur choix
- Un nouveau vote est effectué jusqu’à obtenir un consensus

huit.re/cci-uml

[Lien manager]

Phases de Scrum



Principes Scrum (1/2)

Isolement de l'équipe de développement

. l'équipe est isolée de toute influence extérieure qui pourrait lui nuire. Seules l'information et les tâches reliées au projet lui parviennent : pas d’évolution des besoins dans chaque sprint.

Développement progressif

. afin de forcer l'équipe à progresser, elle doit livrer une solution tous les 30 jours. Durant cette période de développement l'équipe se doit de livrer une série de fonctionnalités qui devront être opérationnelles à la fin des 30 jours.

Principes Scrum (2/2)

Pouvoir à l'équipe

. l'équipe reçoit les pleins pouvoirs pour réaliser les fonctionnalités. C'est elle qui détient la responsabilité de décider comment atteindre ses objectifs. Sa seule contrainte est de livrer une solution qui convienne au client dans un délai de 30 jours.

Contrôle du travail

. le travail est contrôlé quotidiennement pour savoir si tout va bien pour les membres de l'équipe et à la fin des 30 jours de développement pour savoir si la solution répond au besoin du client.

Scrum : rôles responsabilités

- Product owner
  - responsable officiel du projet
- Scrum Master
  - expert de l’application de Scrum
- Scrum Team
  - équipe projet.
- Management
  - prend les décisions

Scrum : pratiques

- Product Backlog
  - état courant des tâches à accomplir
- Effort Estimation
  - permanente, sur les entrées du backlog
- Sprint
  - itération de 30 jours
- Sprint Planning Meeting
  - réunion de décision des objectifs du prochain sprint et de la manière de les implémenter
- Sprint Backlog
  - Product Backlog limité au sprint en cours
- Daily Scrum meeting
  - ce qui a été fait, ce qui reste à faire, les problèmes
- Sprint Review Meeting
  - présentation des résultats du sprint

Exemples

- Cetrea - cetrea.dk
  - Plus loin dans les tests (système pour hopitaux)
- Cozycloud - cozycloud.cc
  - Lean officeless
    - Communication
    - Tout est dématérialisé

Plan

1. Retour rapide sur les méthodes de conception
2. Principes des méthodes Agile
3. XP : eXtreme Programming
4. Scrum
5. **Autres méthodes**

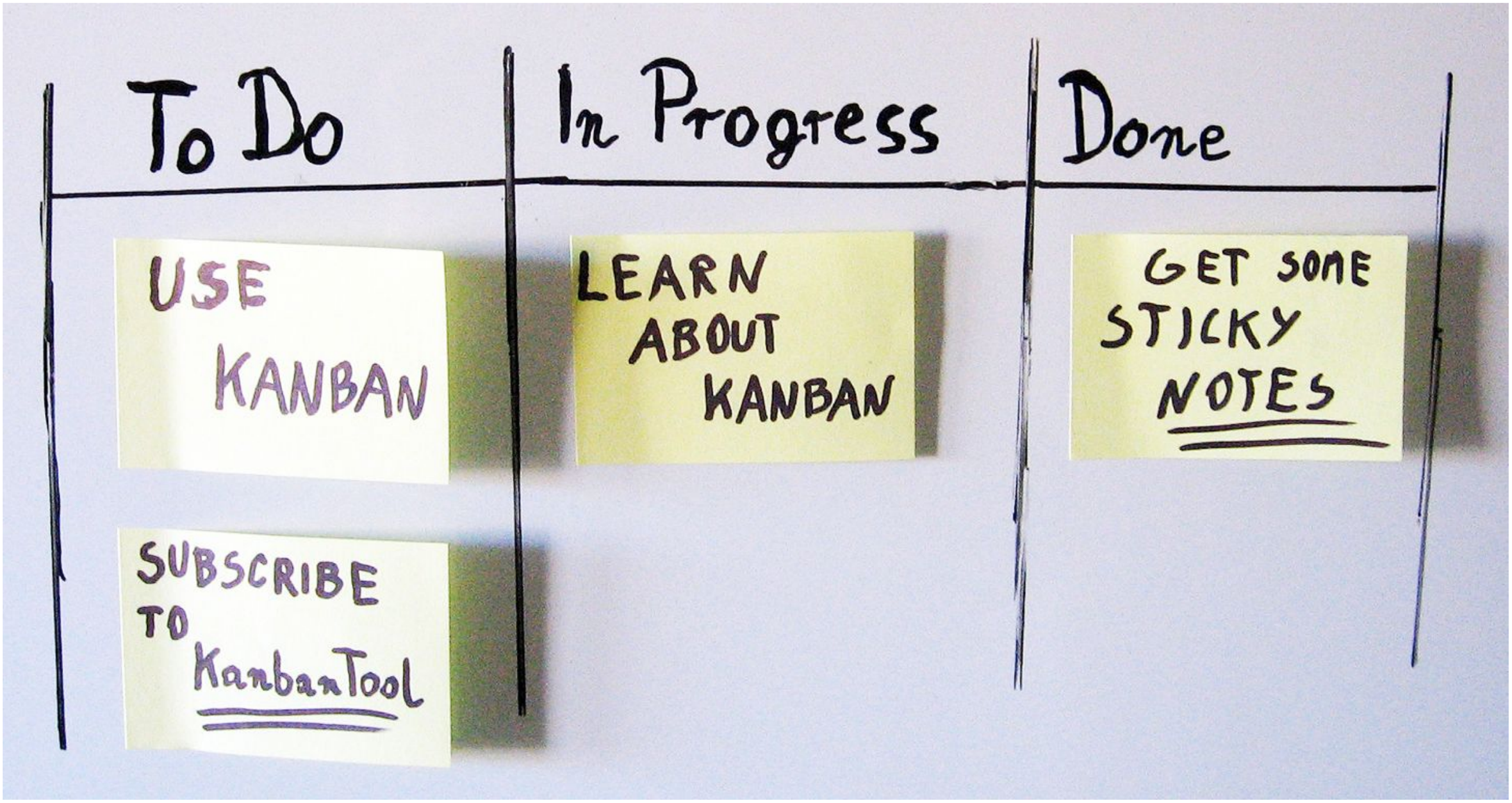
Le lean

- Lean = “sans gras” (mais pas maigre non plus), “au plus juste”
- Méthode de gestion de production de chez Toyota
- S’applique aux développements informatiques agiles
- Se concentre sur l’élimination des gaspillages

Principes du Développement Lean

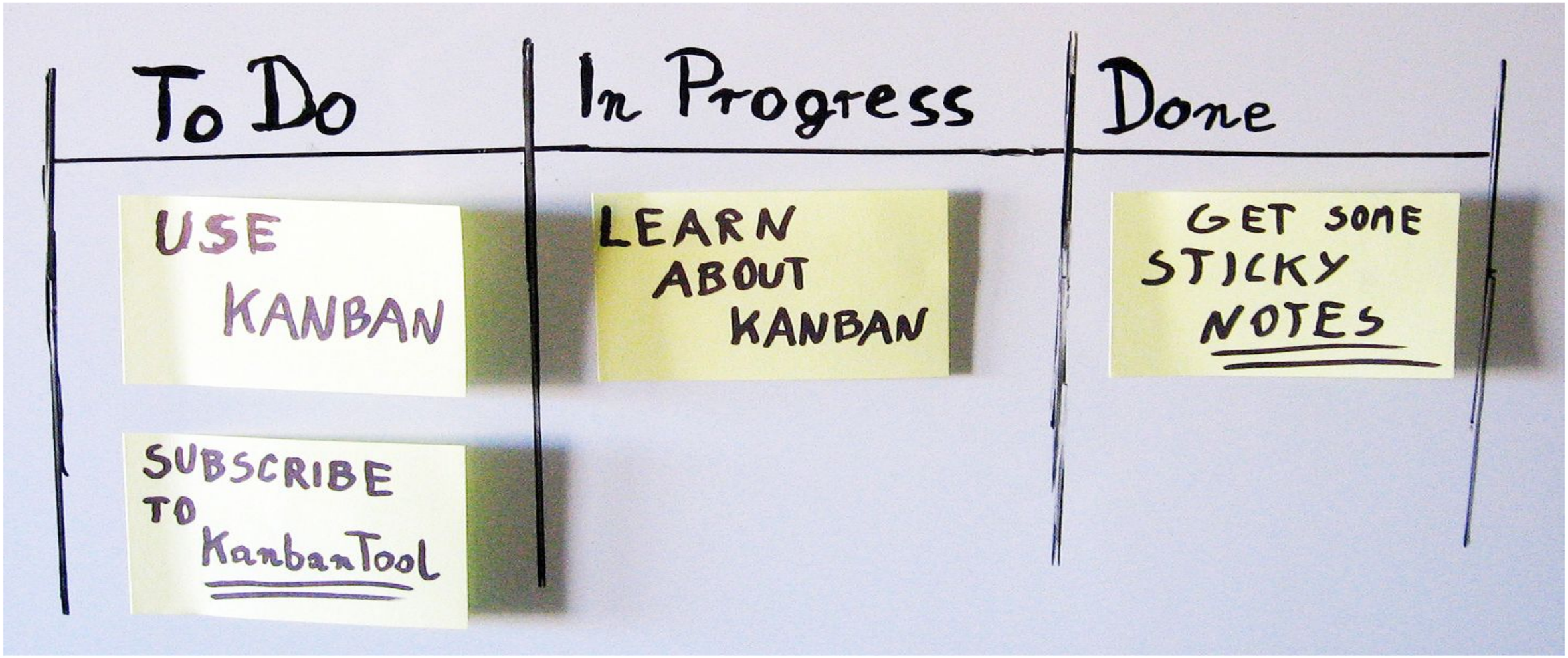
1. Eliminer les gaspillages
2. Améliorer l'apprentissage.
3. Décider aussi tard que possible.
4. Livrer aussi vite que possible.
5. Donner le pouvoir à l'équipe.
6. Intégrer la qualité dès la conception.
7. Considérer le produit dans sa globalité.

Kanban


To Do
In Progress
Done
USE KANBAN
LEARN ABOUT KANBAN
GET SOME STICKY NOTES
SUBSCRIBE TO KanbanTool

Exercice - Kanban

Ajoutez des colonnes à ce Kanban afin qu'il soit plus précis



Kanban, les colonnes classiques

Backlog	Ready	Coding	Testing	Approval	Done

Coders’ dojo

- parallèle arts martiaux / conception-programmation OO
- il faut s’entraîner à appliquer des « routines » connues avant de pouvoir commencer à les utiliser de façon créative, voire à en inventer de nouvelles
- les débutant doivent apprendre des maîtres
- un exemple de formation :
  - http://www.xpday.net/Xpday2005/CodersDojo.html

Conclusion

« *There is no silver bullet* »

. Même si le développement incrémental permet de s'affranchir de beaucoup de problèmes, il y aura quand même des problèmes. Mais ceux-ci seront normalement d'ampleur plus faible, et mieux gérés.

Toute méthode est adaptable et doit être adaptée

. Mais, lorsque l'on débute, il vaut mieux ne pas trop s'écarter de la voie décrite pour bien comprendre au départ (cf. musique)

RAPPEL

Mercredi prochain

Deux groupes

- TD de 9h00 à 12h00
- TD de 14h00 à 17h00