# PROCÉDÉS DE DÉVELOPPEMENT

## Procédés synthétiques (agiles)

PR002
v231c

2017-01-23

Luc LAVOIE et Christina KHNAISSER
Département d'informatique
Faculté des sciences

UNIVERSITÉ DE SHERBROOKE

Luc.Lavoie@USherbrooke.ca
http://info.usherbrooke.ca/llavoie

# Table des matières

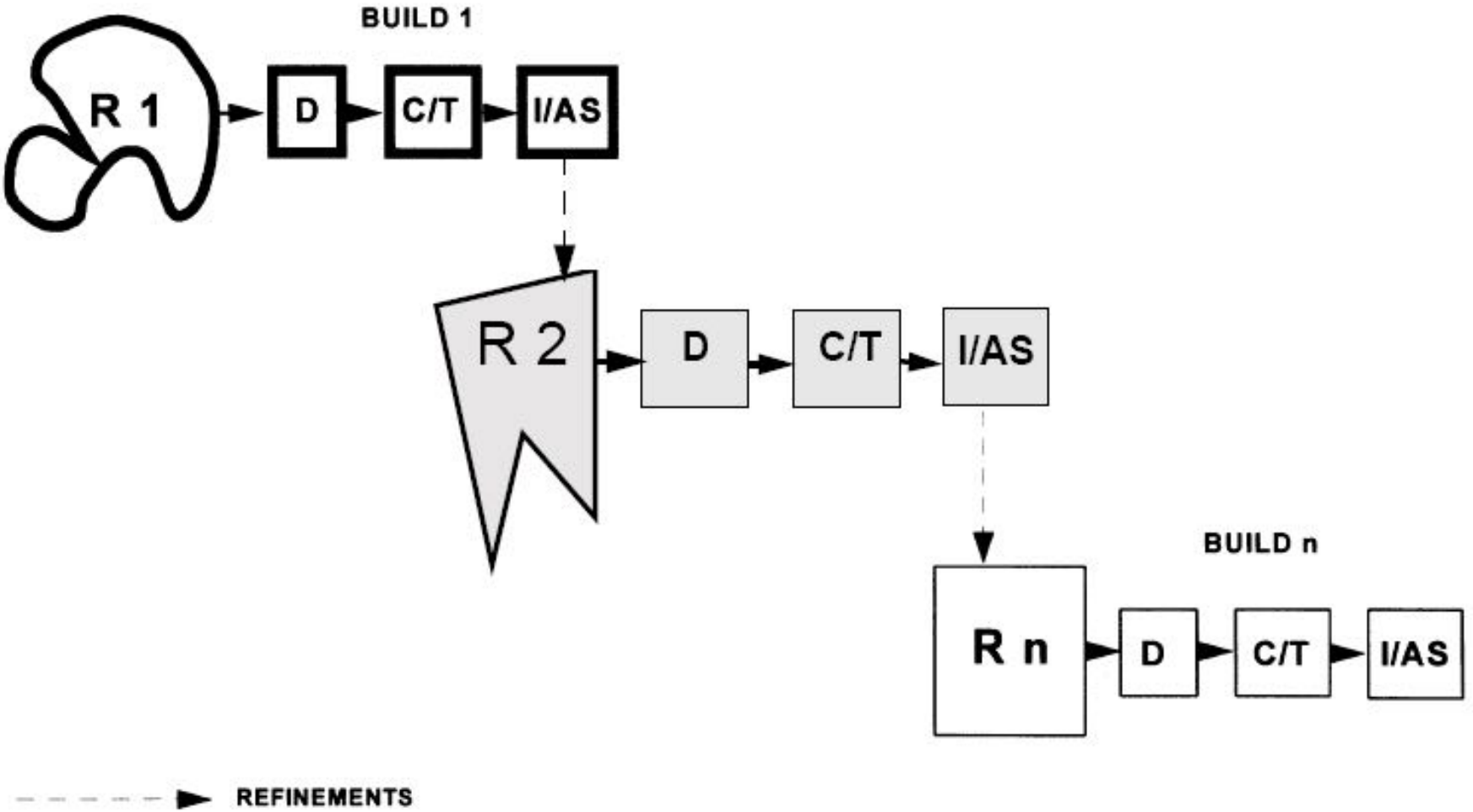
# APERÇU

- Concepts
- Schéma général
- Commentaires

# Aperçu
## Concepts (synthèse)

- Les procédés synthétiques induisent la spécification des exigences plutôt qu'ils n'en découlent.
- L'exploration, l'analyse et la synthèse des exigences se font tout au long des itérations.
- La dernière itération doit voir à leur spécification.
- Tout au long du procédé, la pleine participation des représentants des parties intéressées est requise.

# Aperçu
## Schéma général



# Aperçu
## Commentaires

- C'est une approche empirique tout aussi scientifiquement valable que la méthode prédictive (spéculative) : dans les deux cas, l'expérience tranchera.
- Il est facile de garantir le respect de l'échéancier est du budget.
- Plusieurs caractéristiques corollaires du procédé posent cependant de sérieux problèmes en pratique :
  - incertitude quant à la convergence (en fonctionnalité ou en qualité) ;
  - monopolisation quasi exclusive d'importantes ressources au sein des parties intéressées ;
  - très grande **expertise** requise de l'ensemble des participants ;
  - très grande **expérience** requise de l'ensemble des participants.

# XP

- Présentation
- Itérations XP
- Caractéristiques
- Courte analyse
- Quand l'utiliser ?

# XP

## Présentation

- L'*Extreme Programming* (XP) aurait initialement été mis au point par Kent Beck, Ward Cunningham et Ron Jeffries lors du développement d'un logiciel de calcul des rémunérations chez Chrysler.

- La programmation extrême est une méthode agile constituée d'un ensemble de techniques et d'activités pratiquées par une équipe de programmeurs dans le but de produire des logiciels de qualité.

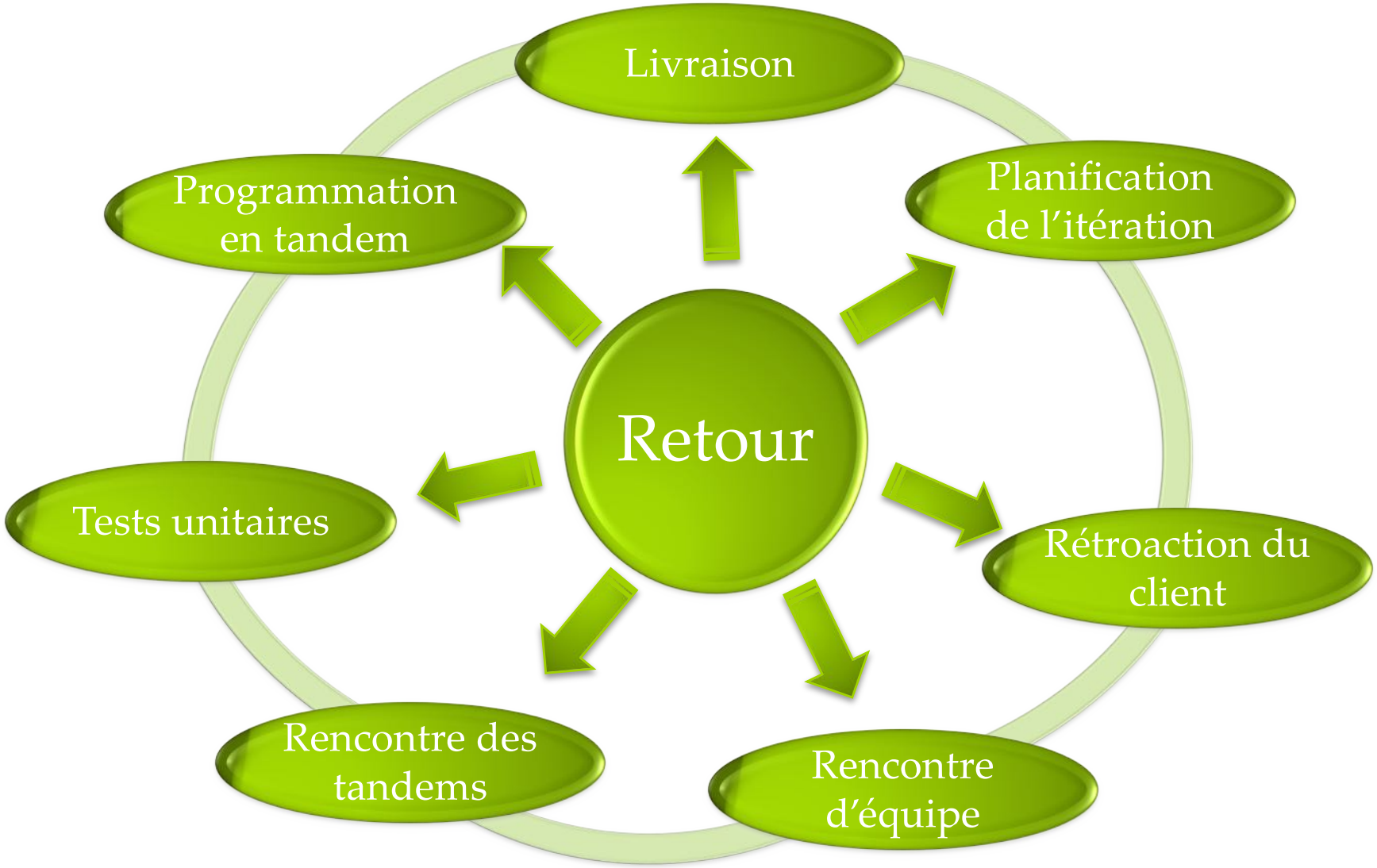
!

# XP

## Caractéristiques

- Programmation en tandem.
- Utilisation des « histoires de cas » (à ne pas confondre avec les cas d'utilisation).
- Développement systématique des essais unitaires avant la conception.
- Primauté de la simplicité de conception sur tout autre critère (de conception).
- Systématisation de la refactorisation.
- Pilotage du projet par le client.

# ITÉRATIONS XP



# XP

## Courte analyse

| Caractéristique | Analyse |
|---|---|
| Tandem | Est-ce vraiment mieux que les autres méthodes ? Aucune étude concluante à ce jour. |
| Histoire de cas | Est-ce vraiment mieux que les cas d'utilisation ? Aucune étude concluante à ce jour. |
| Antériorité des tests unitaires | Reprise d'une bonne idée déjà largement pratiquée, mais non imposée par les autres procédés. |
| Primauté de la simplicité | Exclusion pratique de nombreux domaines d'application. |
| Refactorisation | Excellente idée rendue nécessaire par la primauté de la simplicité, mais qui a des limites : celles de l'architecture. |
| Pilotage par le client | Principe naturel ou abdication de responsabilité par l'informaticien ? |

# XP

## Variantes

- Tandem (variantes diverses)
  - ange gardien en alternance
  - âme soeur
  - pour revue seulement (ce n'est plus du XP!)
- Cycles
  - variables
  - évolutifs (alternance court-moyen)
  - fixes (tendance Scrum)

# XP

## Quand l'utiliser ?

- Équipe réduite
  (maximum 12 personnes, plutôt 4 à 8)
- Durée réduite
  (maximum 1 an, plutôt 4 à 8 mois)
- Besoins changeants
- Architecture fixée à l'avance

# Scrum

- Présentation
- Itérations du Scrum
- Caractéristiques
- Quand l'utiliser ?

http://learning2bdoctor.files.wordpress.com/2011/01/rugby-scrum.jpg

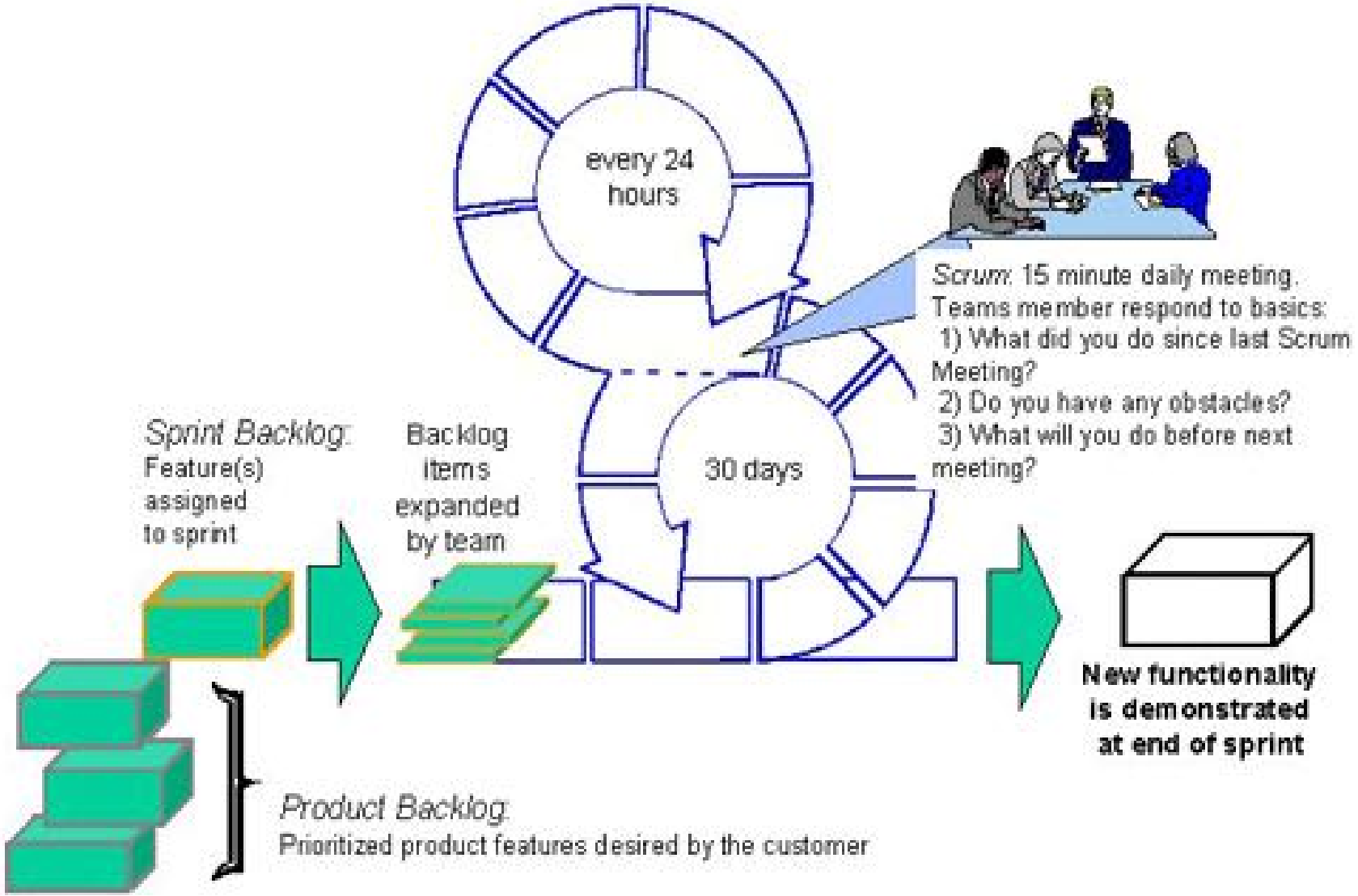
# Scrum
## Présentation

- Le terme *Scrum* emprunté au rugby signifie *mêlée*. Ce processus s'articule en effet autour d'une équipe soudée, qui cherche à atteindre un but... par tous les moyens !

- C'est un processus évolutif qui vise à livrer à chaque cycle
  - une version du produit atteignant *(au moins)* la qualité minimale convenue,
  - construite à partir des seules ressources disponibles,
  - dans une durée déterminée.

# SCRUM

## ITÉRATIONS DU SCRUM




source : Pressman

# Scrum

## Principe de base

### source http://fr.wikipedia.org/wiki/Scrum_(méthode)

- Le principe de base de Scrum est de focaliser l'équipe de façon itérative sur un ensemble de fonctionnalités (*features*) à réaliser, dans des itérations de durée fixe de une à quatre semaines, appelées **sprints**.
- Chaque sprint possède un **but** à atteindre, défini par le *directeur de produit*, à partir duquel sont choisies les fonctionnalités à implémenter dans ce sprint.
- Un sprint aboutit toujours sur la livraison d'un produit partiel fonctionnel.
- Pendant ce temps, le *ScrumMaster* a la charge de réduire au maximum les perturbations extérieures et de résoudre les problèmes non techniques de l'équipe.

# Scrum

## Vocabulaire

### source http://fr.wikipedia.org/wiki/Scrum_(méthode)

**Backlog**

C'est une liste des exigences et des fonctions que le client désire avoir. Des nouvelles données peuvent être ajoutées à la liste en tout moment.

**Product Backlog**

Liste des fonctionnalités priorisées par le client.

**Sprints :**

Unité de travail nécessaire pour accomplir une exigence définie dans le Backlog dans une durée déterminée (4 semaines, 30 jours, etc.). Durant un sprint les exigences développées, ne sont jamais modifié.

**Scrum meeting :**

Les rencontres quotidiennes sont très courtes (15 minutes, debout). Elles consistent principalement à répondre aux questions suivantes (pour chaque participant):

- Quels sont les travaux accomplis depuis la dernière rencontre?
- Quels sont les problèmes rencontrés?
- Que doit-on accomplir avant la prochaine rencontre?

**Demos**

Cette activité commence par livrer l'incrément pour que les fonctionnalités soient approuvées et évaluées par le client.

# Scrum

## Caractéristiques

- L'équipe est dirigée par le client à temps complet, le client étant aussi un expert du domaine d'application.
- L'équipe est vue comme une extension du client qui prend toutes les décisions.
- Pas de planification.
- L'intégration se fait par accumulation.
- Un objectif est déterminé à chaque itération, et un seul.
- Compilation quotidienne (voire continue) et livraison quotidienne.

# Scrum
## Corollaires

- Environnement de développement sophistiqué, intégrant vérifications automatisées, tests unitaires, taux de couverture, etc. :

  *si ça « build », c'est bon !!!*

- L'intégration doit pouvoir se faire par simple accumulation (donc l'architecture **pré-établie** doit être **très** simple).
- Les programmeurs sont des *Code wariors* (on oublie les définitions de Wirth, de Dijkstra, de Knuth et de Boehm et coll.)

# Scrum

## Quand l'utiliser ?

- Petite équipe (max. 8, plutôt 4 à 6).
- Courte durée (max. 1 an, plutôt 3 à 6 mois).
- À considérer lors du développement d'applications éphémères reposant sur une expertise pointue.
  - Développement de jeux.
  - Développement de certains types de sites et d'applications web.

# Scrum
## En conclusion ?

- Cohérent avec l'approche Unix, telle que mise de l'avant par Kernignan et Plauger
  - *Try brute force, think later.*
  - *If brute force don't work, you're not using enough.*
  - *If it still don't work, call Chuck Norris.*

http://www.gargarismes-ergonomiques.com/allblogs/public/article/melee-scrum.jpg

# KA(N?)BAN

## En développement

- Origines
- Adaptation au développement logiciel
- Commentaires

# Kanban

## Origines

- Procédé d'origine industriel visant à optimiser la production en chaîne.
- Orienté par le « just-in-time » et la minimisation de l'en-cours.
- Construit autour du concept de lot.
- Mettant de l'avant la constitution de petites unités de production autonomes.
- Utilisant le modèle de qualité TQM plutôt que le modèle ISO.
- Expérimenté (avec succès) par Toyota.

- Et pour le logiciel ?

# Kanban
## Adaptations

### 4 Principes

- Partir avec l'existant
- Opérer uniquement des changements évolutifs et un seul à la fois
- Respecter intégralement le processus, les rôles et les pratiques courantes
- Encourager le leadership à tous les niveaux

### 6 pratiques

- Visualiser
- Limiter l'en-cours
- Gérer les flux
- Rendre explicites toutes les règles
- Intégrer la rétroaction dans chaque activité
- Améliorer collectivement, évoluer expérimentalement et seulement sur une base scientifique

# Synthèse

**Et le gagnant est...**

- Processus dérivé de la spirale
- XP
- Scrum
- Kanban
- ?

## Comment réagissez-vous à ceci ?

| Facteur | SP | XP | SC |
|---|---|---|---|
| **Gestion** | | | |
| Simplicité | A | B | A+ |
| Expérience requise | G | M | M+ |
| Implication du client | DC | PR | CO |
| Suivi contractuel | A+ | A | B |
| **Projet** | | | |
| Taille | M-G | P-M | P |
| Durée totale optimale | B | B+ | A+ |
| Variabilité de la durée totale | B | B+ | A |
| Coût global optimal | C | B | A- |
| Variabilité du coût total | B+ | C | A |
| **Tolérance aux variations** | | | |
| Exigences | A+ | B- | A |
| Ressources humaines | A- | B+ | A |
| Risques technologiques | B | B+ | A+ |
| **Produit fini** | | | |
| Modifiabilité | A+ | A- | A |
| Qualité architecturale | A | C+ | B |
| Qualité globale | A | A- | B+ |
| Correctitude | B+ | A | C |
| Validité | A- | A+ | A |
| Adéquation | A- | B | A+ |

### Synthèse

SP : Spirale

XP : Programmation extrême

SC: Scrum

--

A : Excellent

B : Très bien

C : Bien

D : Passable

E : Insuffisant

--

PI :
ponctuelle, initiale mais intensive

PR :

ponctuelle, répétée mais pas toujours intensive

DR :

discontinue, répétée et d'intensité variable

DC :

discontinue, répétée et d'intensité variable

CO :

continuelle

--

M : moyen

G : grand

P : petit

• : toutes tailles

--

? : pas de consensus

## Et à ceci ? Faites votre propre grille !

| Facteur | SP | XP | SC |
|---|---|---|---|
| **Gestion** | | | |
| Simplicité | B | B | A+ |
| Expérience requise | M+ | G | G |
| Implication du client | DC | PR | CO |
| Suivi contractuel | A+ | A | B |
| **Projet** | | | |
| Taille | M-G | P-M | P |
| Durée totale optimale | B | B+ | NA |
| Variabilité de la durée totale | B | B+ | NA |
| Coût global optimal | B+ | B | NA |
| Variabilité du coût total | B+ | C | NA |
| **Tolérance aux variations** | | | |
| Exigences | A | A- | A |
| Ressources humaines | A- | A+ | C |
| Risques technologiques | A+ | B+ | A+ |
| **Produit fini** | | | |
| Modifiabilité | A+ | B | C- |
| Qualité architecturale | A+ | B | C |
| Qualité globale | A | B | C |
| Correctitude | A | A | C |
| Validité | A | B+ | B |
| Adéquation | A | A | A |

## Synthèse

SP : Spirale

XP : Programmation extrême

SC: Scrum

--

A : Excellent

B : Très bien

C : Bien

D : Passable

E : Insuffisant

--

PI :
ponctuelle, initiale mais intensive

PR :

ponctuelle, répétée mais pas toujours intensive

DR :

discontinue, répétée et d'intensité variable

DC :

discontinue, répétée et d'intensité variable

CO :

continuelle

--

M : moyen

G : grand

P : petit

- : toutes tailles

--

? : pas de consensus

# Synthèse
## XP vs Scrum

### XP

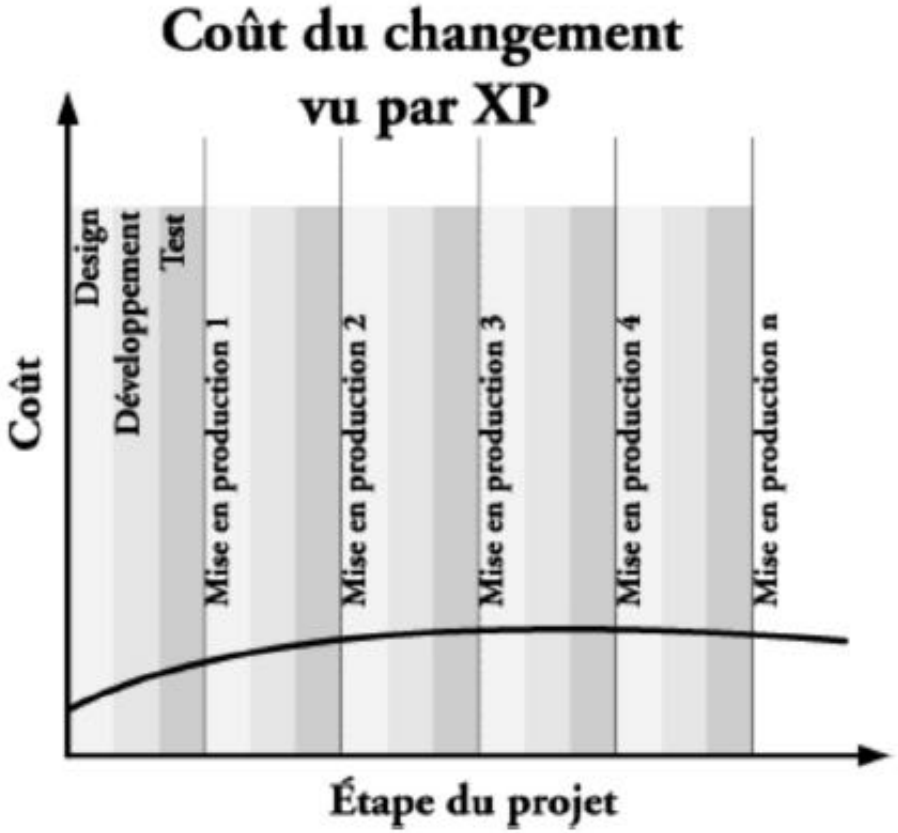
- Travail en tandem
- Cas d'utilisation au démarrage
- Planification KISS
- Itération typique de 2 semaines
- Développement des essais avant la conception
- Conception par prototypage
- Codage des tests unitaires avant celui des modules

### Scrum

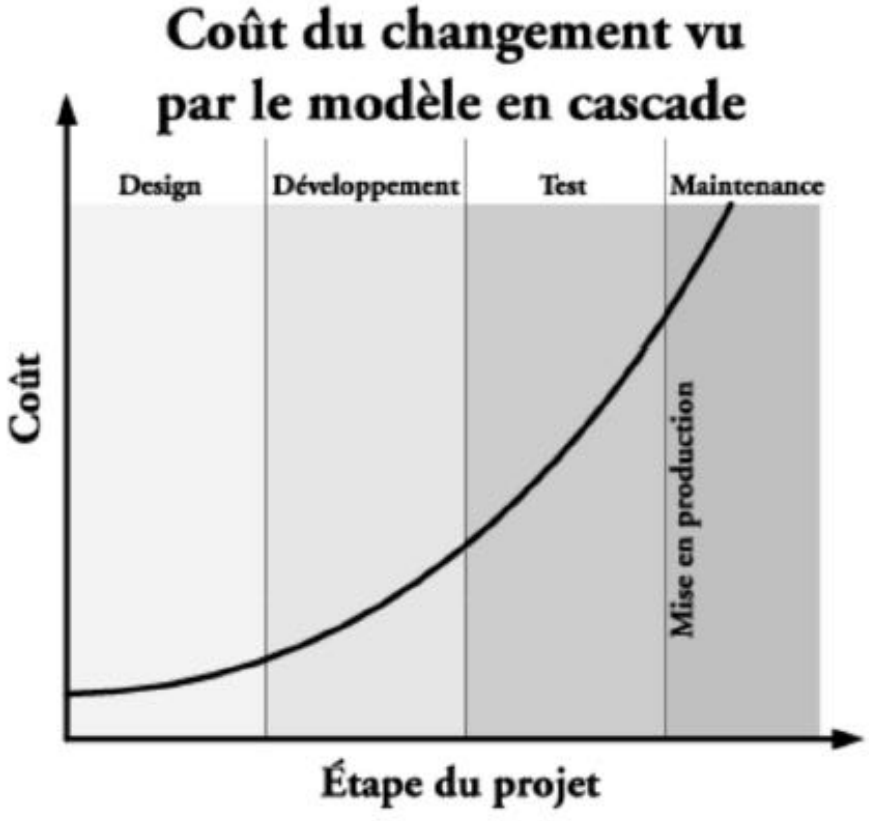
- Travail en symbiose
- Un objectif fonctionnel à la fois
- Peu de planification
- 10 à itérations de 24 heures par cycle de « stabilisation »
- Livraison interne à la fin de chaque itération
- Livraison externe à la fin de chaque cycle
- « Scrum » tous les matins (15 à 30 minutes)
- Développement par effet tunnel

# XP vs Cascade
## Coût du changement

### XP




### Cascade



# Agilité
## Concepts

- L'agilité n'est pas tant un procédé qu'une façon de l'appliquer.

- Un processus synthétique est « agile » lorsque l'équipe qui le réalise adhère au manifeste agile et en pratique les 12 adages.

- Le texte des deux diapositives suivantes est tiré de http://agilemanifesto.org.
(sauf les annotations en orange)

# Agilité
## Manifeste

« Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

- Les individus et leurs interactions plus que les processus et les outils.
- Des logiciels opérationnels plus qu'une documentation exhaustive.
- La collaboration avec les clients plus que la négociation contractuelle.
- L'adaptation au changement plus que le suivi d'un plan.

Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers. »

# Agilité

## Commentaires

« Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

- Les individus et leurs interactions plus que les processus et les outils. *Les processus agiles sont ceux qui requièrent et consomment les plus d'outillage.*
- Des logiciels opérationnels plus qu'une documentation exhaustive. *Comment maintenir l'adéquation, la responsabilité et l'imputabilité.*
- La collaboration avec les clients plus que la négociation contractuelle. *Oui, tant que tout le monde il est beau et gentil.*
- L'adaptation au changement plus que le suivi d'un plan. *Plus communément, n'appelle-t-on pas cela la fuite en avant?*

Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers. »

# Agilité
## Adages

- Objectif :
  Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Règle :
  Welcome changing requirements, even late in development.
- Affirmation (non démontrée) :
  Agile processes harness change for the customer's competitive advantage.
- Règle :
  Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Règle :
  Business people and developers must work together daily throughout the project.
- Règle :
  Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- Affirmation (non démontrée) :
  The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Axiome :
  Working software is the primary measure of progress.
- Objectif :
  Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Affirmation (non démontrée) :
  Continuous attention to technical excellence and good design enhances agility.
- Affirmation (non démontrée) :
  Simplicity - the art of maximizing the amount of work not done - is essential.
- Affirmation (non démontrée) :
  The best architectures, requirements, and designs emerge from self-organizing teams.
- Règle :
  At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

# VOCABULAIRE USUEL

- Risques
- Essais d'acceptation
- Essais et tests d'intégration
- Essais et tests de régression

# RÉFÉRENCES

- Pressman, chapitre 3 et 4
- Leffingwell, chapitre 3