

# Panorama du Développement Classique au Développement Agile



*Isabelle BLASQUEZ*  
*@iblasquez*

*Septembre 2016*

# Principales phases du développement logiciel

c  
o  
n  
c  
e  
implémentation

Qualification ou Recette

validation

n

i

y

test

e

d  
mise en production

a  
i  
t  
e  
n  
a  
n  
c  
e  
c  
u  
m  
e  
n  
t  
a  
t  
i  
o  
n

Déploiement ou Livraison

# A propos de la phase d'Analyse des exigences (Requirements phase)

**La phase d'analyse des exigences** est la période du cycle de vie pendant laquelle les exigences, fonctionnelles et non fonctionnelles du produit logiciel, sont définies et documentées. (**IEEE, 1990<sup>1</sup>**).

Ce que doit faire le logiciel

Cette phase donne lieu à l'écriture d'un document de **spécifications** qui précise les missions du logiciel. Ce document est une trace des besoins utilisateurs et sera utilisé dans les autres phases du cycle de développement.

<sup>1</sup> IEEE Standard Glossary of Software Engineering Terminology

**Abstract:** IEEE Std 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, identifies terms currently in use in the field of Software Engineering. Standard definitions for those terms are established.  
**Keywords:** Software engineering; glossary; terminology; definitions; dictionary

# A propos de la phase de Conception (Design phase)

COMMENT  
faire ce logiciel

**La phase de conception** est la période du cycle de vie pendant laquelle l'architecture logicielle, les composants logiciels, les données et les interfaces sont conçus et documentés afin de satisfaire aux exigences.  
**(IEEE, 1990).**

# A propos de la phase d'Implementation (Implementation phase/Coding)

Ecriture  
du code

**La phase d'implémentation** est la période du cycle de vie pendant laquelle le logiciel est créé et débuggé à partir des spécifications de conception. (IEEE, 1990).

Les tâches de cette phase se concentrent autour du **code** où les composants sont implémentés et testés individuellement **dans un langage de programmation** donné afin de mettre en œuvre la conception

# A propos de la phase de Test (Test phase)

Qualité logicielle

**La phase de Test** est la période du cycle de vie consacrée à l'intégration et à l'évaluation des composants et du logiciel afin de vérifier les exigences aussi bien au niveau système qu'utilisateur (**IEEE, 1990**).

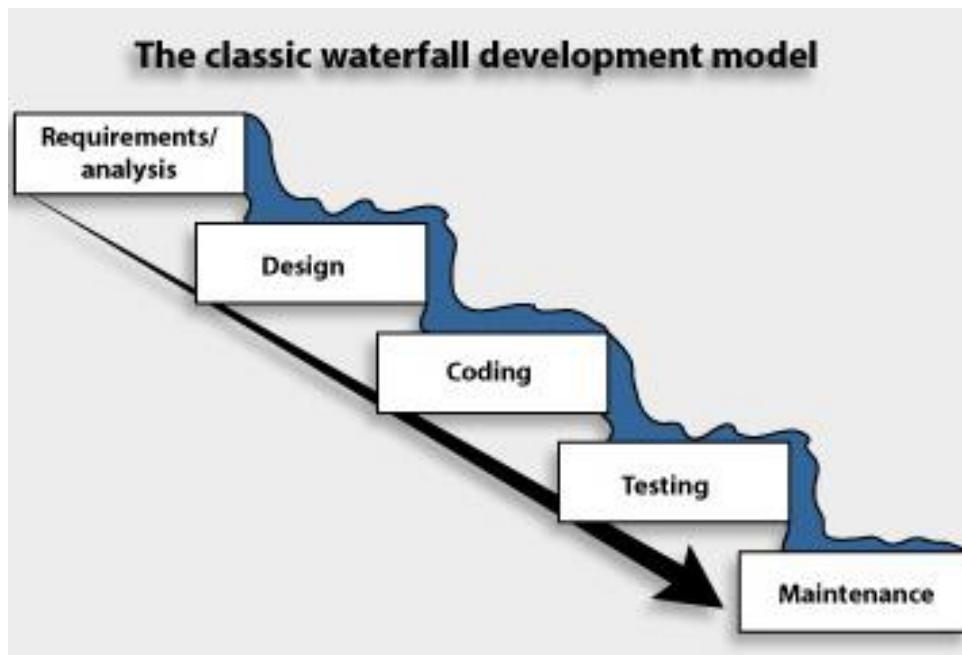
# Modèle de développement dit « en cascade »

Analyse

Conception

Implémentation

Test

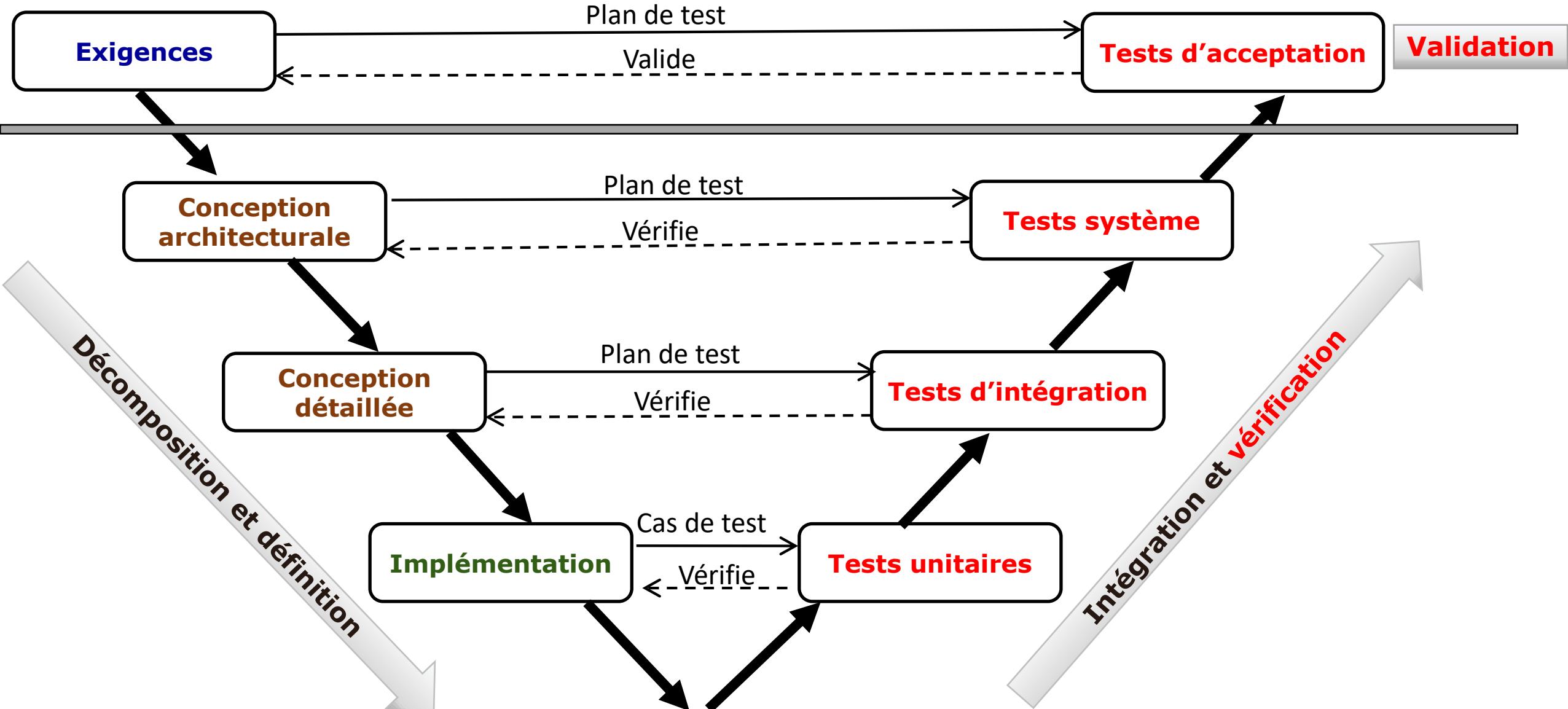


Attention !!! A l'origine, « contre-exemple » utilisé par Royce pour lui permettre d'introduire les bonnes pratiques de développement

Article original de Royce : <http://www.serena.com/docs/agile/papers/Managing-The-Development-of-Large-Software-Systems.pdf>

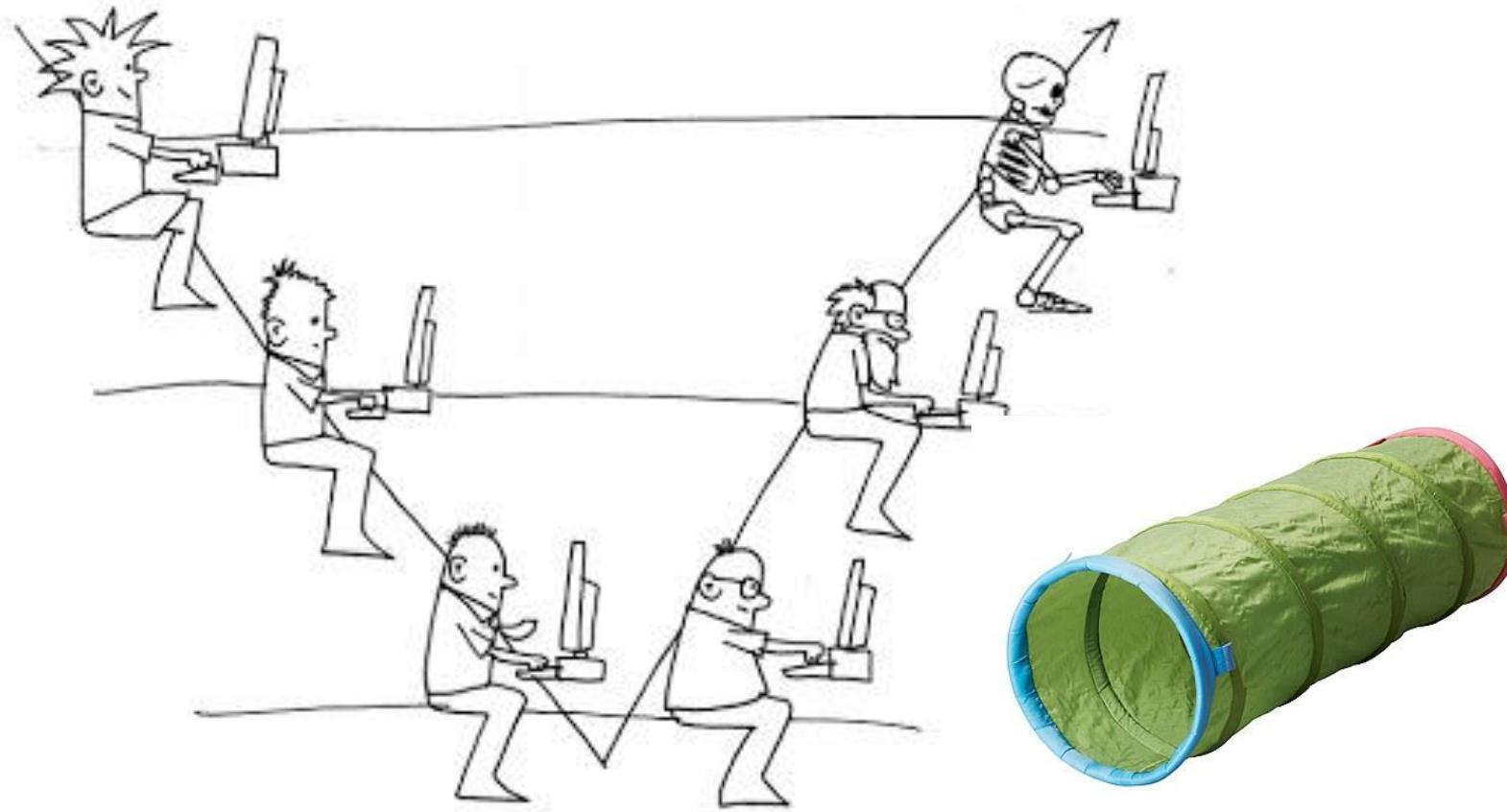
Traduction française : <http://www.fabrice-aimetti.fr/dotclear/public/traductions/waterfall-FR.pdf>

# Cycle en V : standard de l'industrie logicielle

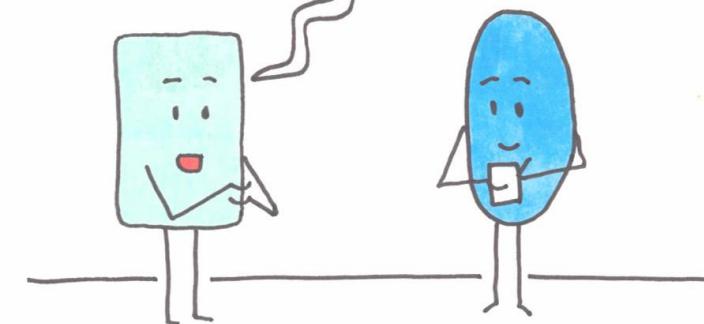


# Un cycle trop long, bien souvent un effet tunnel...

PREMIER JOUR DE STAGE...

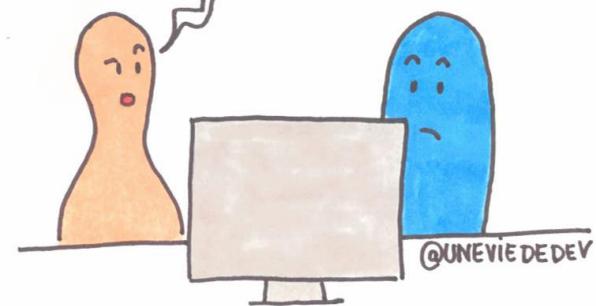


POUR ÊTRE SUR DE RÉPONDRE AUX BESOINS DE L'UTILISATEUR TU VAS FAIRE UN PLAN QUALITÉ, UN DOCUMENT DE SPÉCIFICATION FONCTIONNELLE ET TECHNIQUE, LE PLAN DE VALIDATION ET ENFIN LE DÉVELOPPEMENT. UNE FOIS L'APPLI PRÊTE, TU LE PRÉSENTES À L'UTILISATEUR.



DERNIER JOUR DE STAGE...

MAS CE N'EST PAS DU TOUT CE QUE JE VOULAI



@UNEVIEDEDEV

# Et « Les joies du code » dans tout ça ?

En suivant ces cycles de vie, on peut passer plus de temps sur la façon de développer un système que sur le développement lui-même...

Quand on me dit que  
je vais pouvoir coder  
après 3 mois passés  
sur les plans de tests



<http://lesjoiesducode.fr/post/95172054995/quand-on-me-dit-que-je-vais-pouvoir-coder-apr%C3%A8s-3-mois-pass%C3%A9s-sur-les-plans-de-tests>

Quand je peux enfin lancer  
les tests unitaires  
sur mon appli



<http://lesjoiesducode.fr/post/75046110363/quand-je-peux-enfin-lancer-les-tests-unitaires-sur-mon>

Isabelle BLASQUEZ - 2016

# « Les joies du client » : *Right product ?*



<http://lesjoiesducode.fr/post/98960240314/quand-je-respecte-les-specs-du-client-a-la-lettre>

Quand je respecte les spécifications du client à la lettre...



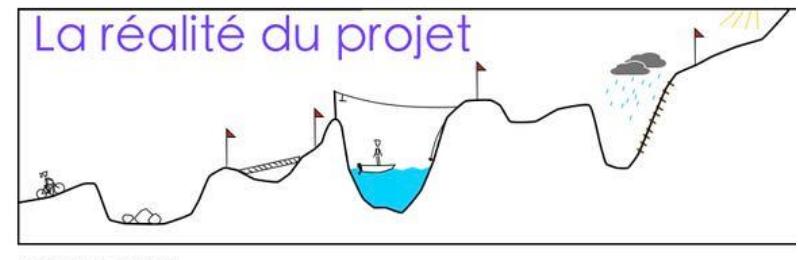
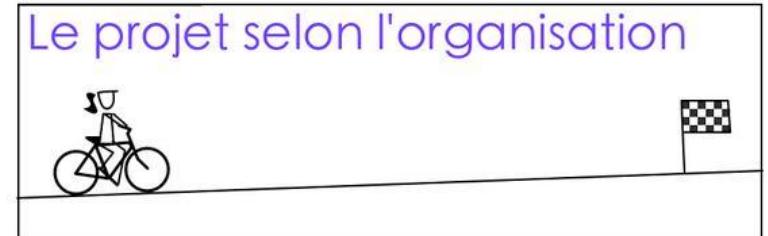
CommitStrip.com

<http://www.commitstrip.com/fr/2014/10/15/when-i-thoroughly-follow-the-requirements/>

Isabelle BLASQUEZ - 2016

# Prise en compte du risque ?

- ✓ **Un risque** peut être défini comme la probabilité qu'un événement, un danger, une menace ou une situation arrive, et que les conséquences indésirables qui en découlent constituent un problème potentiel.



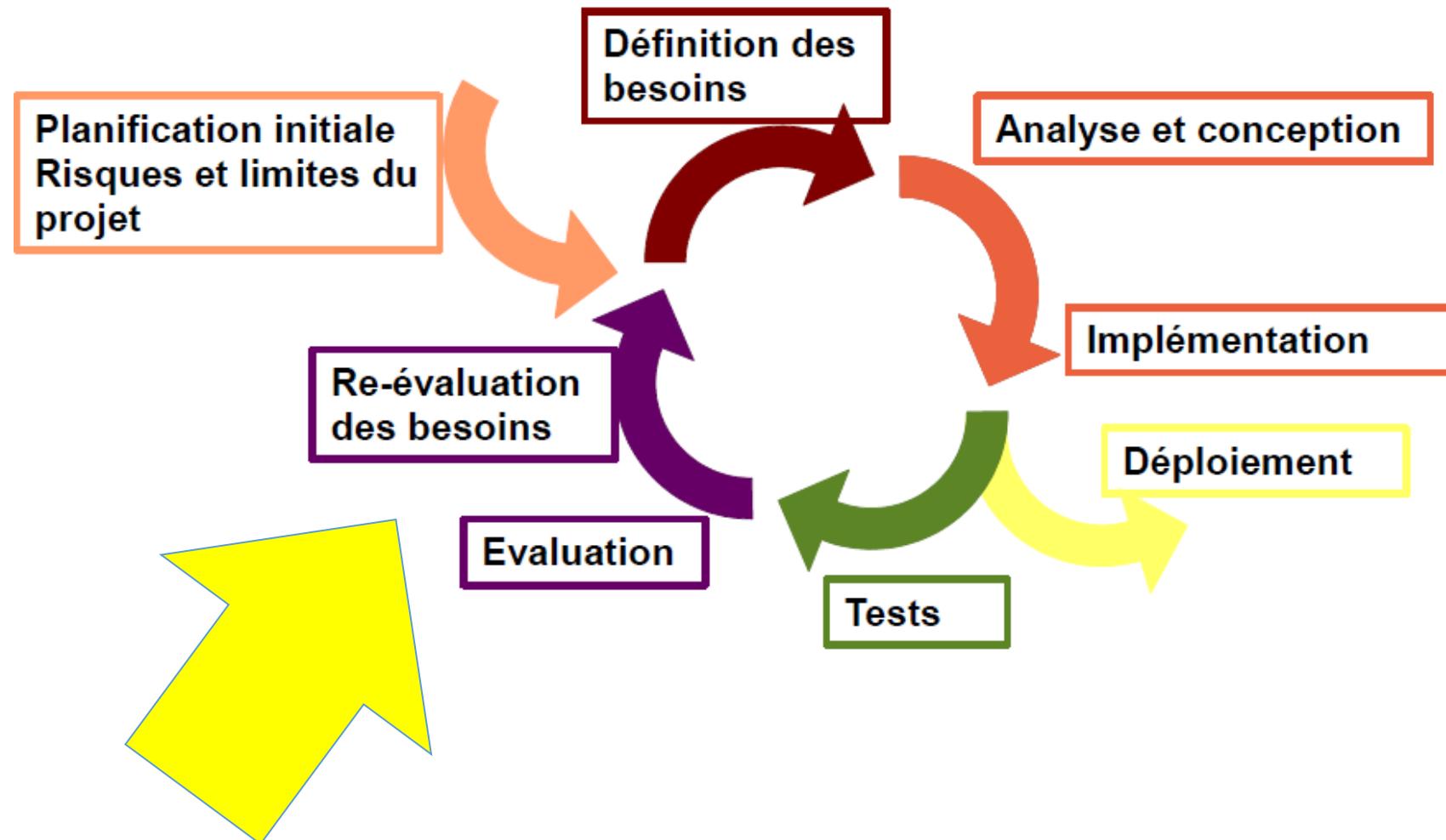
DOGHOUSEDIARIES

Source : <https://twitter.com/agilep/status/398831282051772416/photo/1>

- ✓ 2 types de risques :
  - **risques liés au produit** lié à la **qualité** du produit...
  - **risques liés au projet** qui menacent la capacité de ce dernier à atteindre ses objectifs:
    - les **facteurs organisationnels**
    - les **problème techniques**

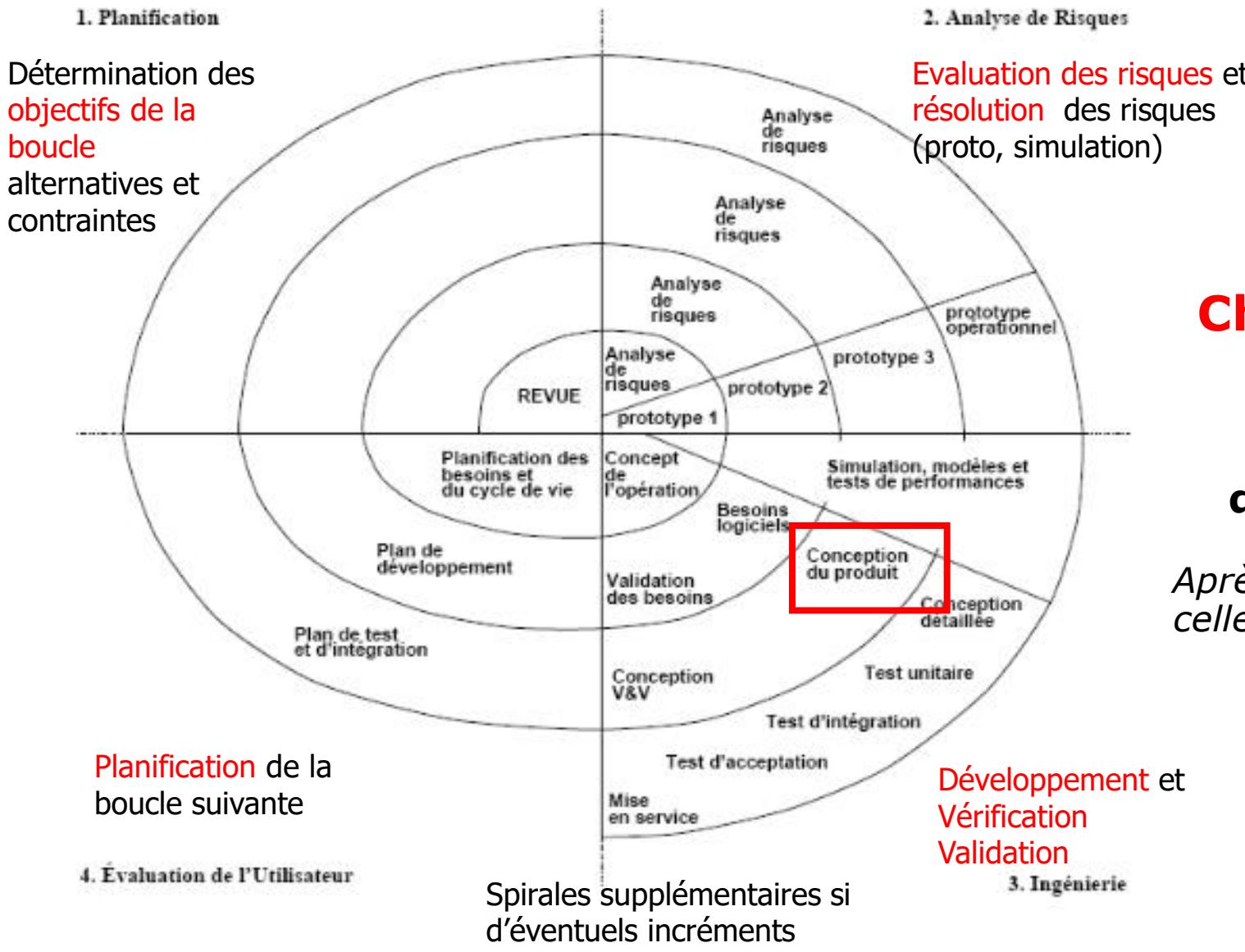


# Un développement **itératif** pour une meilleure évaluation et prise en compte des risques



# Modèle en spirale (Boehm)

Axé sur **l'analyse des risques**, principe **itératif**



**Chaque boucle représente une phase du développement (boucle de faisabilité, boucle de prototypage, des boucles de développement,...)**

Après avoir défini les objectifs et les alternatives, celles-ci sont évaluées par différentes techniques (**prototypage, simulation**, ...), l'étape est réalisée et la suite est planifiée

# Illustration des principes *itératif* et *incrémental*

## ✓ Développement itératif

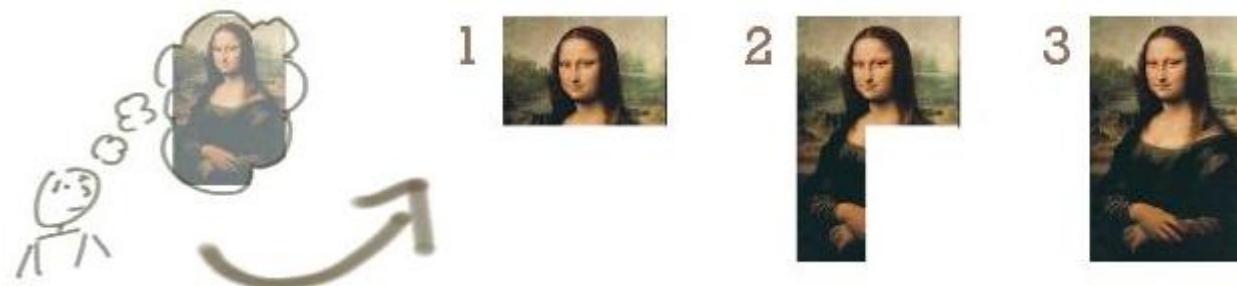
⇒ améliorer un existant pour atteindre un résultat plus satisfaisant



⇒ Permet d'améliorer la **qualité** (affiner, **enrichir**)

## ✓ Développement incremental

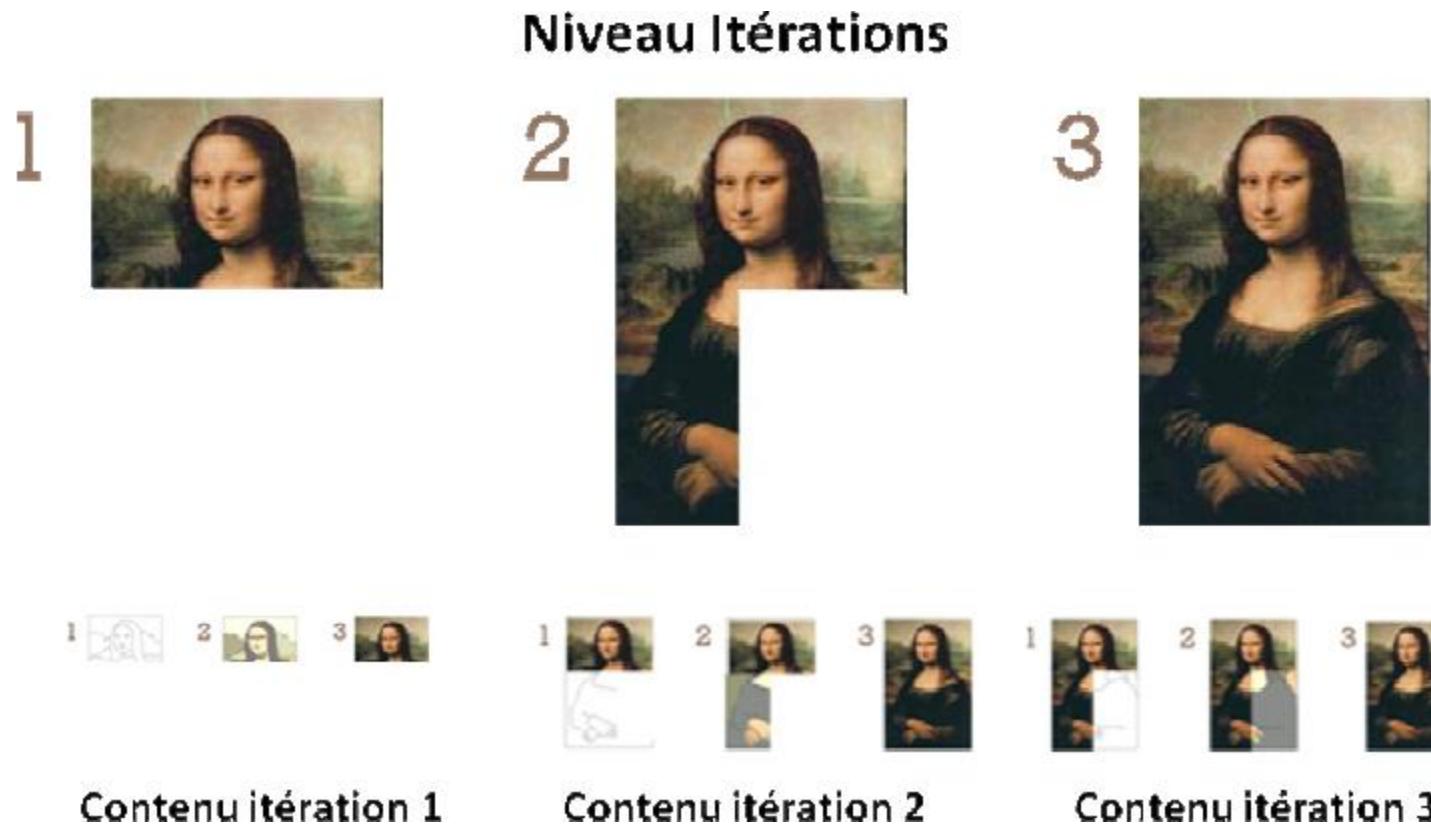
⇒ ajouter pour atteindre le résultat final



⇒ Permet d'améliorer le processus et d'**ajuster** les exigences à l'évolution de l'environnement.  
Analogie : mur en brique

# Démarche *itérative* et *incrémentale*

A chaque nouvelle itération on **ajuste** ce qui a déjà été réalisé et on l'**enrichit** en traitant un spectre fonctionnel plus large.



# Processus Unifié (**UP** : Unified Processus 1997)

- ✓ **Un processus UP** est un processus générique qui définit les exigences suivantes :
  - conduit par les **besoins des utilisateurs**
  - itératif et incrémental
  - centré sur l'architecture
- ...mais aussi...
  - piloté par les risques** (les risques doivent être identifiés au plus tôt et levés rapidement, ce qui doit déterminer l'ordre des itérations)
  - **basé le formalisme UML**
- ✓ RUP (Rational Unified Process) est l'une des plus célèbres implémentations de la méthode UP

En savoir plus : [https://fr.wikipedia.org/wiki/Unified\\_process](https://fr.wikipedia.org/wiki/Unified_process)

[http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251\\_bestpractices\\_TP026B.pdf](http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf)

# RUP (Rational Unified Process)

RUP est organisé autour de **4 phases** et défini en **9 disciplines**.

Contrairement au cycle en V, les disciplines ne sont pas purement séquentielle, mais se répartissent sur plusieurs itérations (itérative & incrémentale)

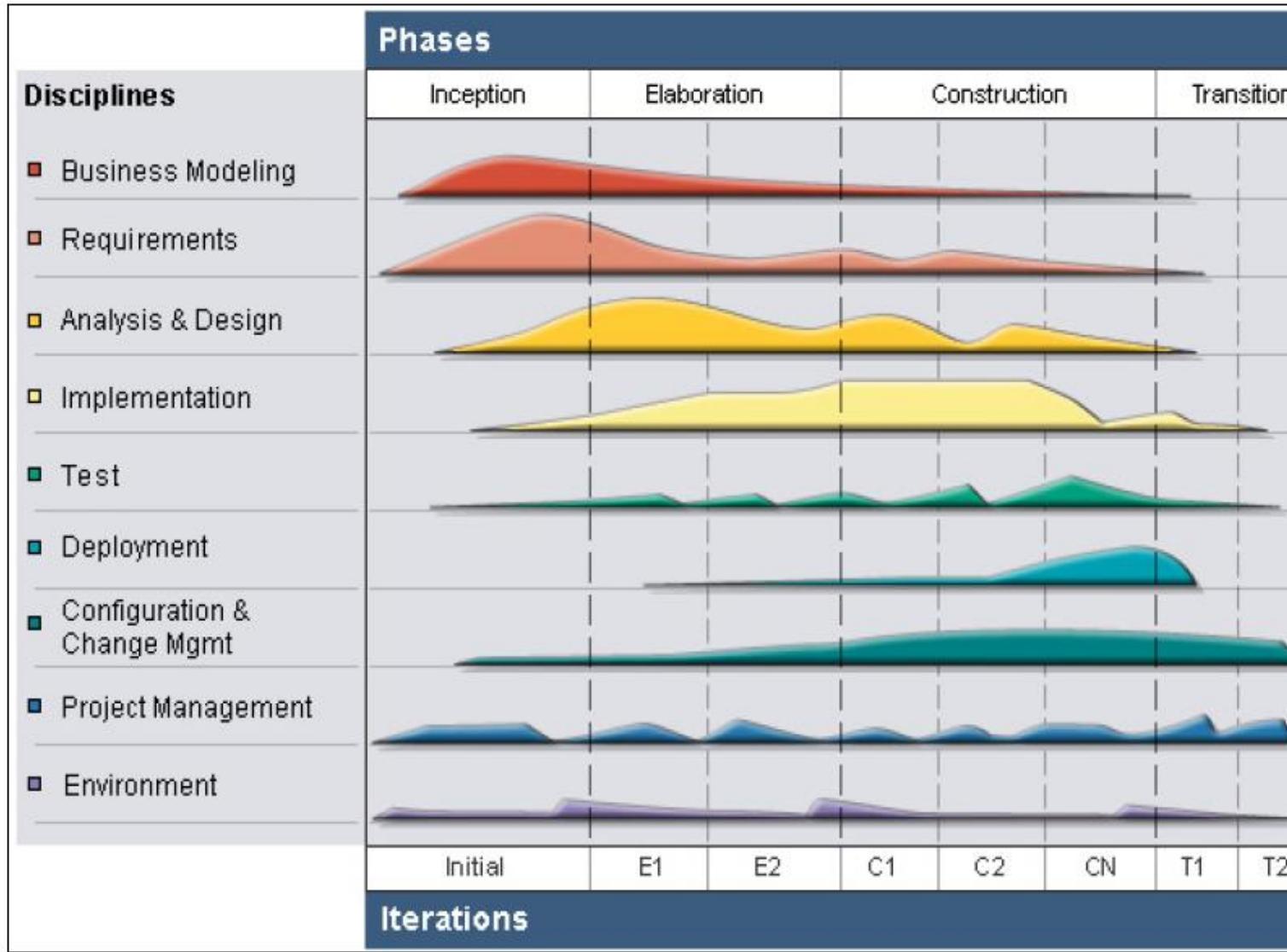


Figure 2-1 Overall Architecture of RUP

extrait de :  
The IBM Rational Unified Process for System z  
A consulter pour tout savoir sur le RUP disponible en ligne sur  
<http://www.redbooks.ibm.com>

# Inconvénient du RUP

## Beaucoup de documentation



Trop d'acteurs isolés  
Et peu de communication  
entre eux ...

### I.I PROJET INFORMATIQUE

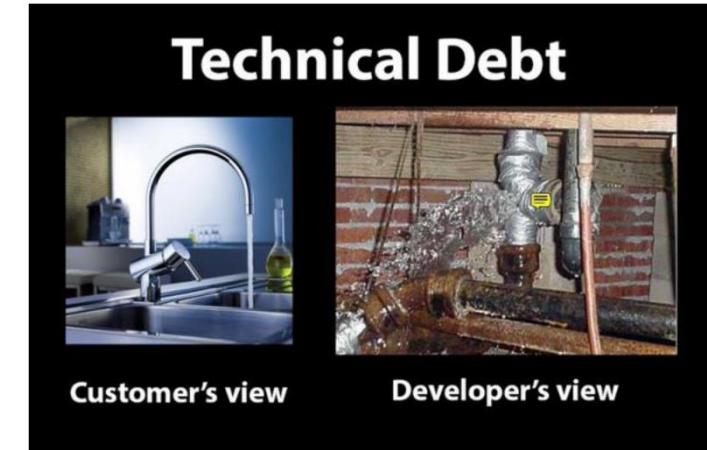


Très axé processus ...

Place tardive pour le code et la technologie

Just tried to explain technical debt to a customer, had to pull this out again...

© Voir la traduction



# ... et voilà *How projects really works*



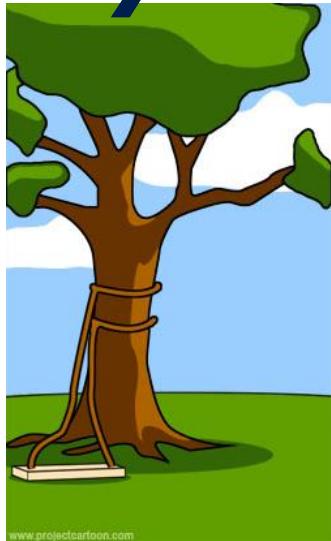
Ce que  
le client  
a expliqué



Ce que  
le chef de projet a  
compris



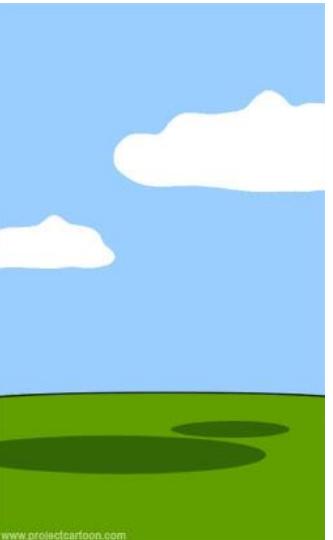
Ce que  
l'analyste l'a conçu



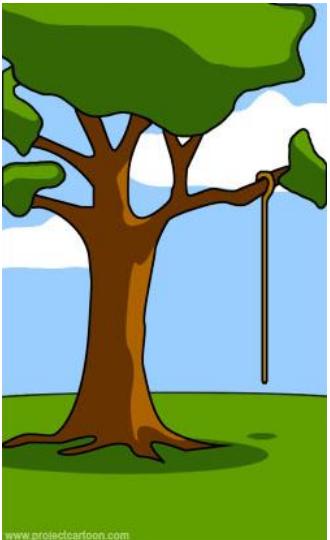
Ce que  
le développeur  
a fait



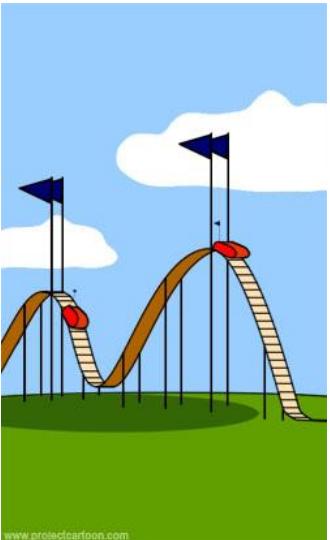
Comment  
les commerciaux  
l'ont décrit



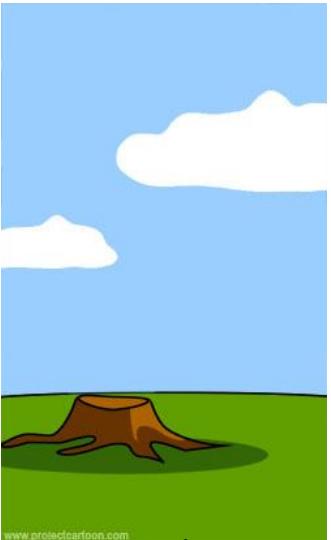
Comment  
le projet a été  
documenté



Ce que la production a  
installé chez le client



Ce que  
le client a été  
facturé



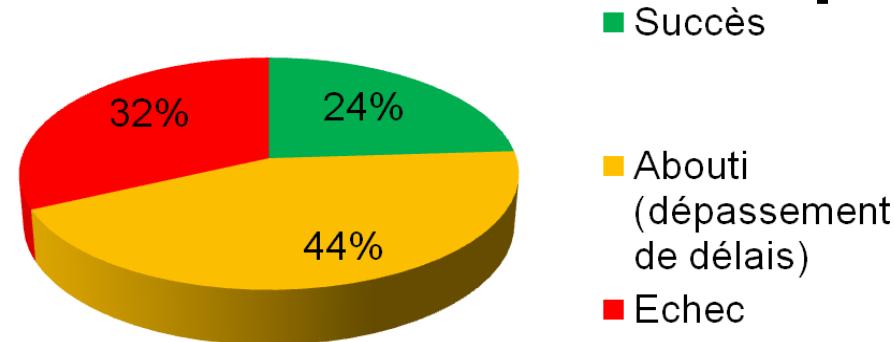
Comment le support  
technique est effectué



Ce dont le client  
avait réellement  
besoin

# ***Un Constat ...***

## **✓ Un fort taux d'échec des projets ...**



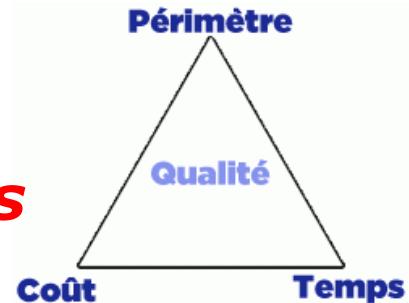
***Taux de succès d'un projet***  
*Données issues du rapport*  
**Chaos 2009 du *Standish Group***

***... et une grande partie de ce qui est fait n'est pas utilisé...***

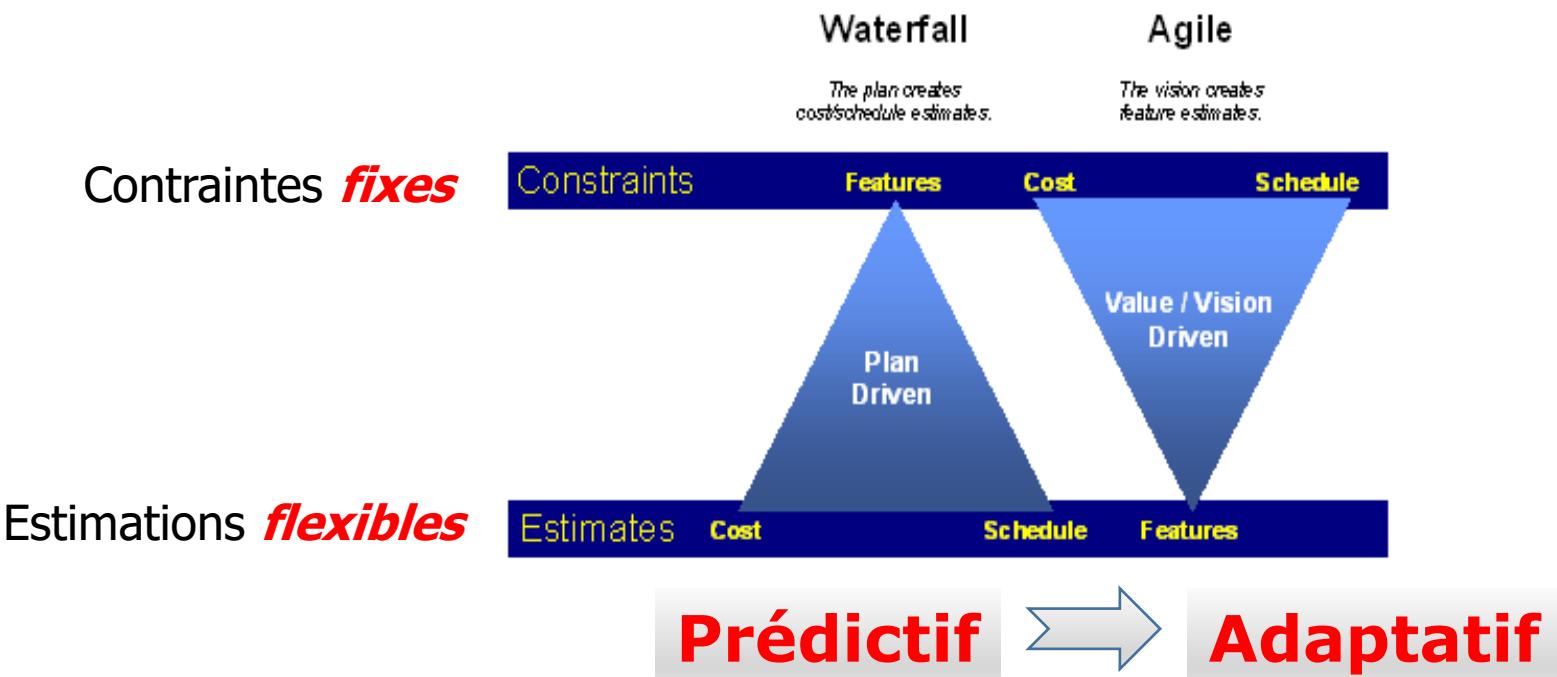
**STOP !**

# A la recherche d'un nouveau paradigme ...

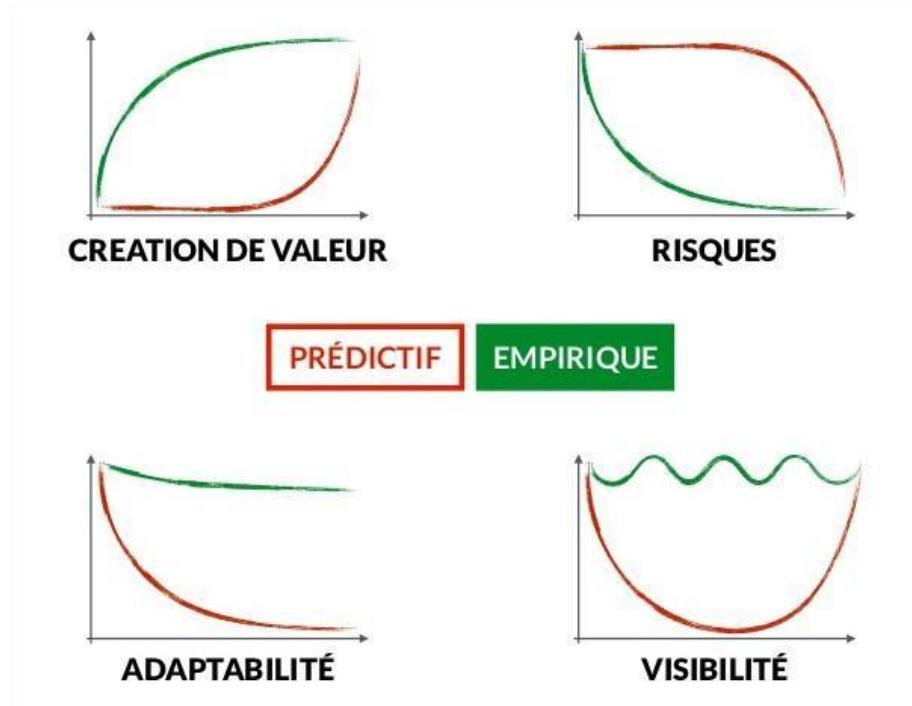
L'objectif d'un processus de développement est de produire des logiciels de **qualité qui répondent aux besoins des utilisateurs dans des temps et des coûts prévisibles**...



⇒ Inverser le triangle de projet pour mieux gérer les contraintes en donnant plus d'importance à la valeur métier plutôt qu'aux prévisions initiales



# Prédicatif vs Adaptatif (empirique)



Extrait original: <http://www.versionone.com/Agile101/Agile-Software-Development-Benefits/>  
Repris en français : <http://fr.slideshare.net/rvignes/lentreprise-agile-29135486>



## Agile Versus Traditional Waterfall

Metric	Waterfall	Agile
Planning scale	Long-term	Short-term
Distance between customer and developer	Long	Short
Time between specification and implementation	Long	Short
Time to discover problems	Long	Short
Project schedule risk	High	Low
Ability to respond quickly to change	Low	High

Extrait : <http://www.venveo.com/blog/agile-vs-waterfall-project-management>

**Empirique :** lorsque le processus doit être expérimenté puis adapté à l'environnement de travail

# En 2001, le Manifeste pour le développement Agile de Logiciels



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

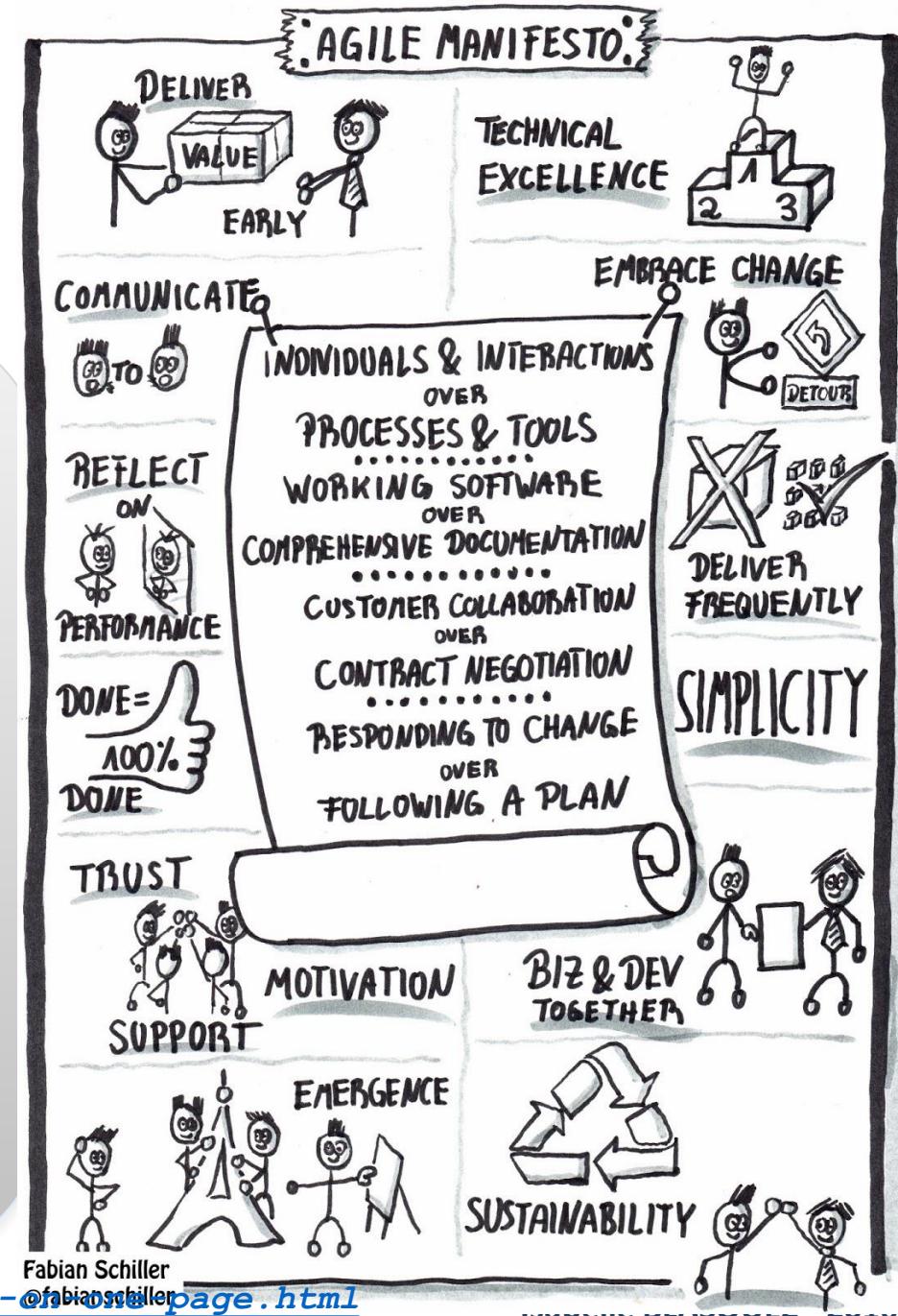
Nous reconnaissons la valeur des seconds éléments,  
mais privilégions les premiers.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

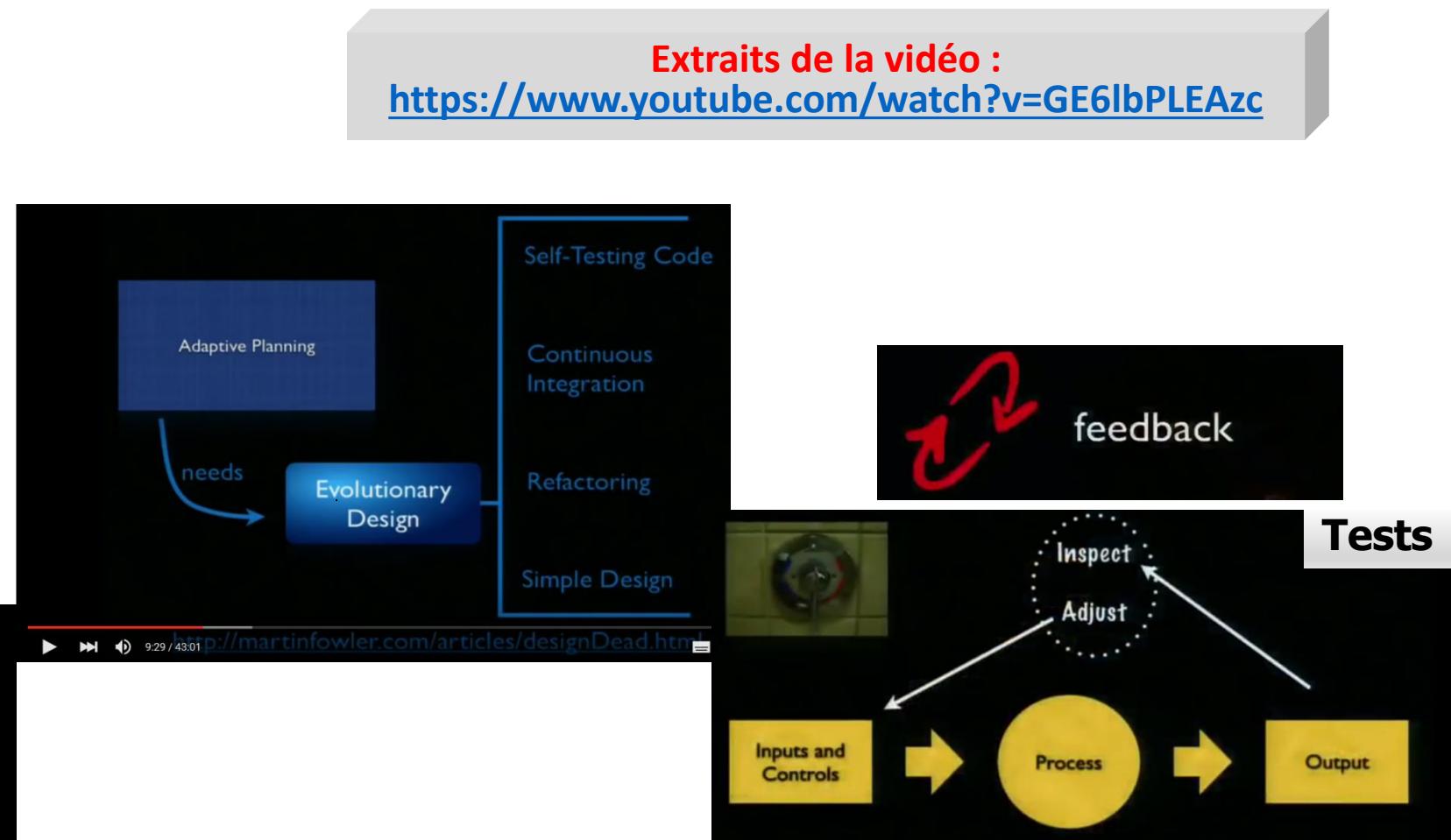
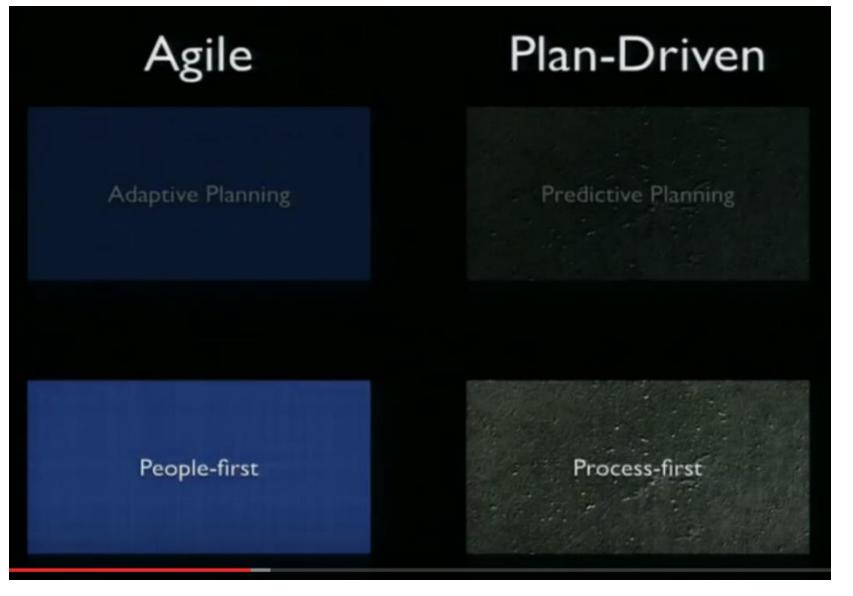
# Les Principes sous-jacents au manifeste

12 principes découlent de ce manifeste.  
Ils indiquent comment implémenter l'Agilité :

- ✓ Livrer de manière **continue** des **fonctionnalités complètes**, utiles et utilisables
- ✓ Développer par **itérations courtes** (quelques semaines)
- ✓ Etre adepte du **changement**, privilégier les solutions **simples** et **adaptables**
- ✓ **Rapprocher** utilisateurs et développeurs
- ✓ **S'améliorer** continuellement



# Explaining Agile - Martin Fowler and Neil Ford at USI



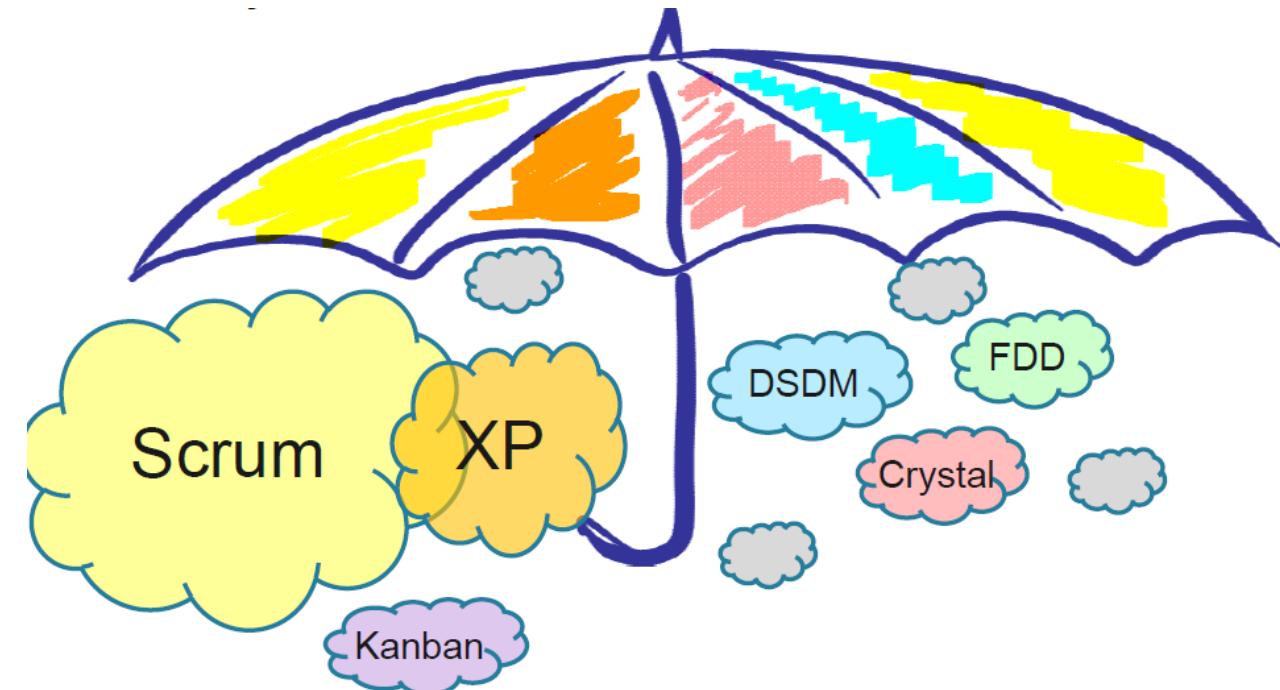
**Pair programming**

communication > technology

Retrospective :  
how to change  
regularly process

In software development, “perfect” is  
a verb, not an adjective

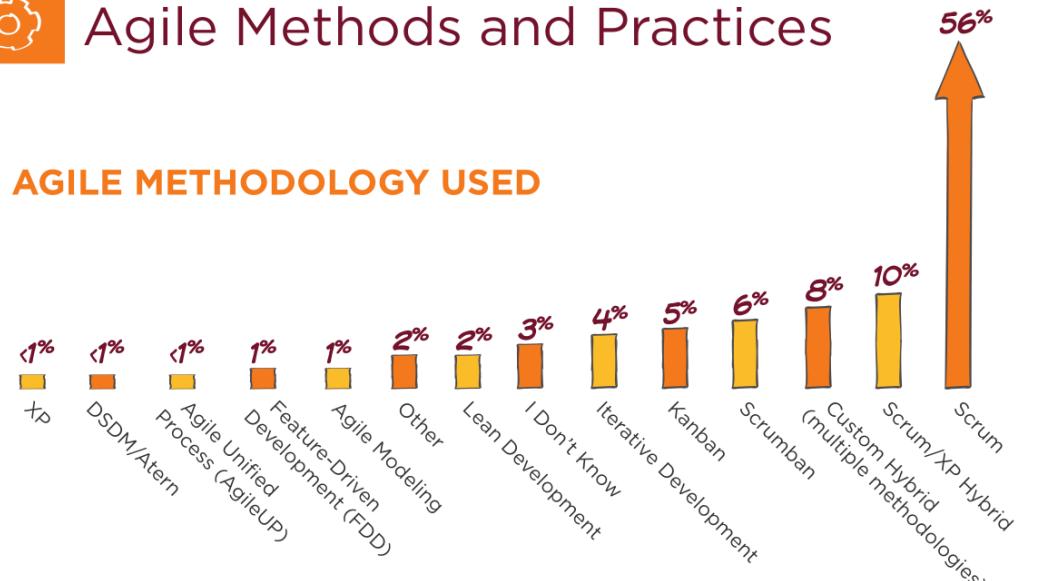
# Des méthodes agiles ...



STATE OF AGILE

## Agile Methods and Practices

### AGILE METHODOLOGY USED



Méthodes agiles les plus utilisées en 2015 (USA et Europe) :

Extrait : <https://www.versionone.com/pdf/state-of-agile-development-survey-ninth.pdf>

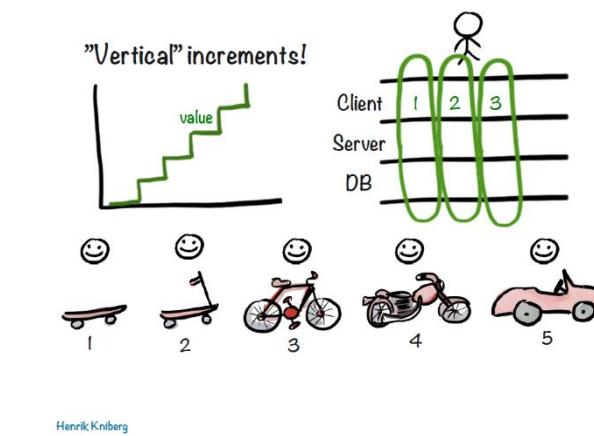
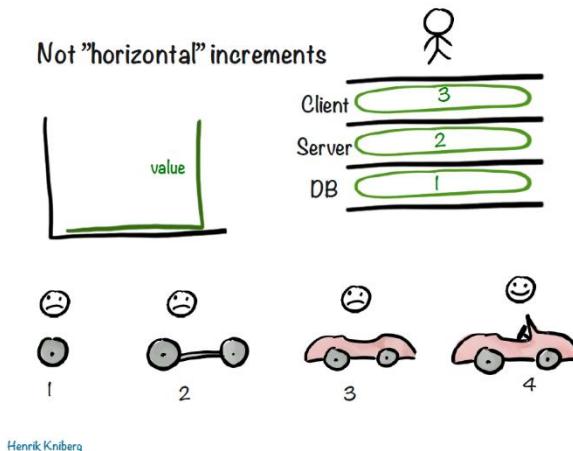
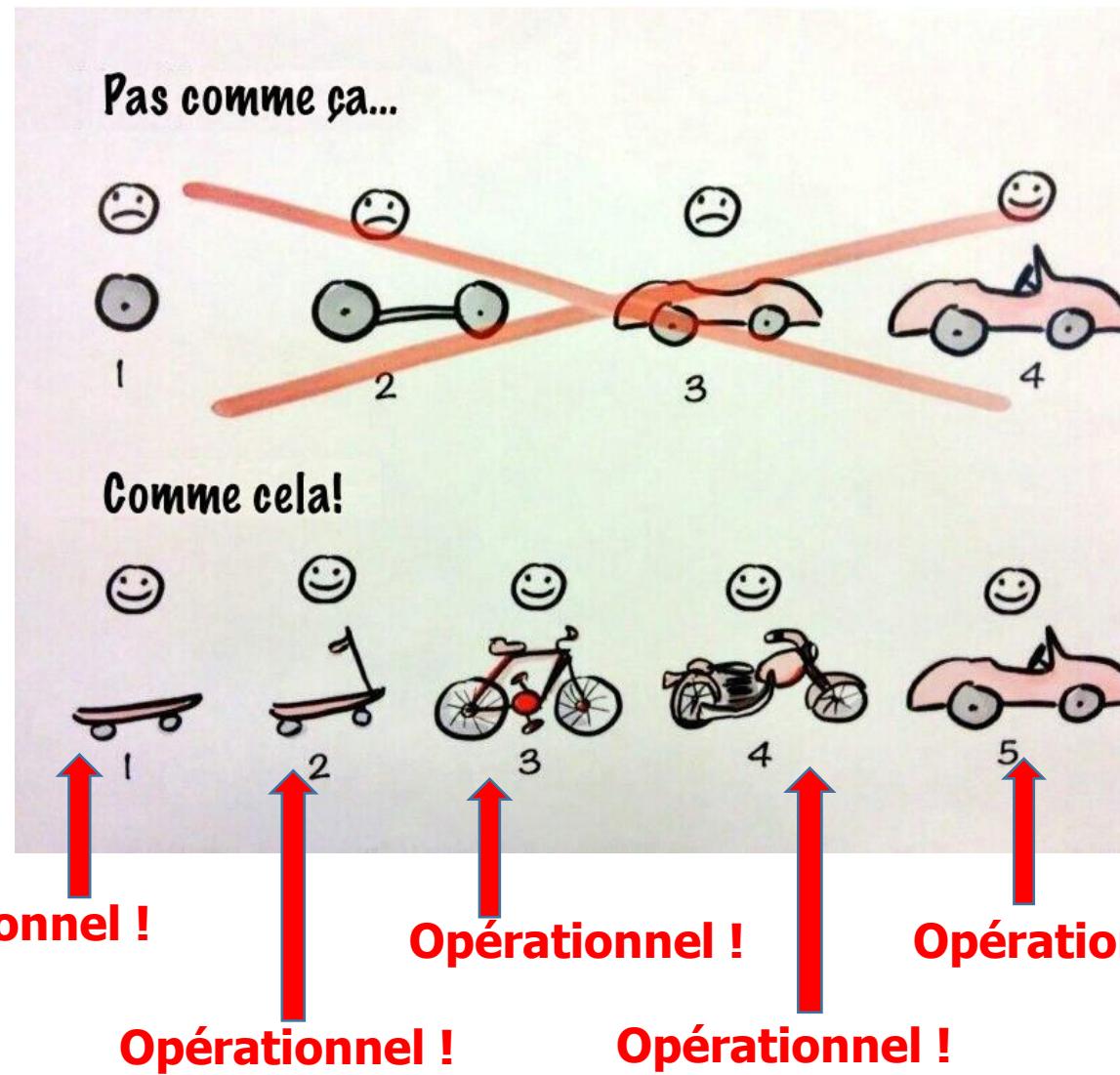
MAIS ATTENTION ...



**David J Bland**  
@davidjbland

Scrum - XP = Agile Theater

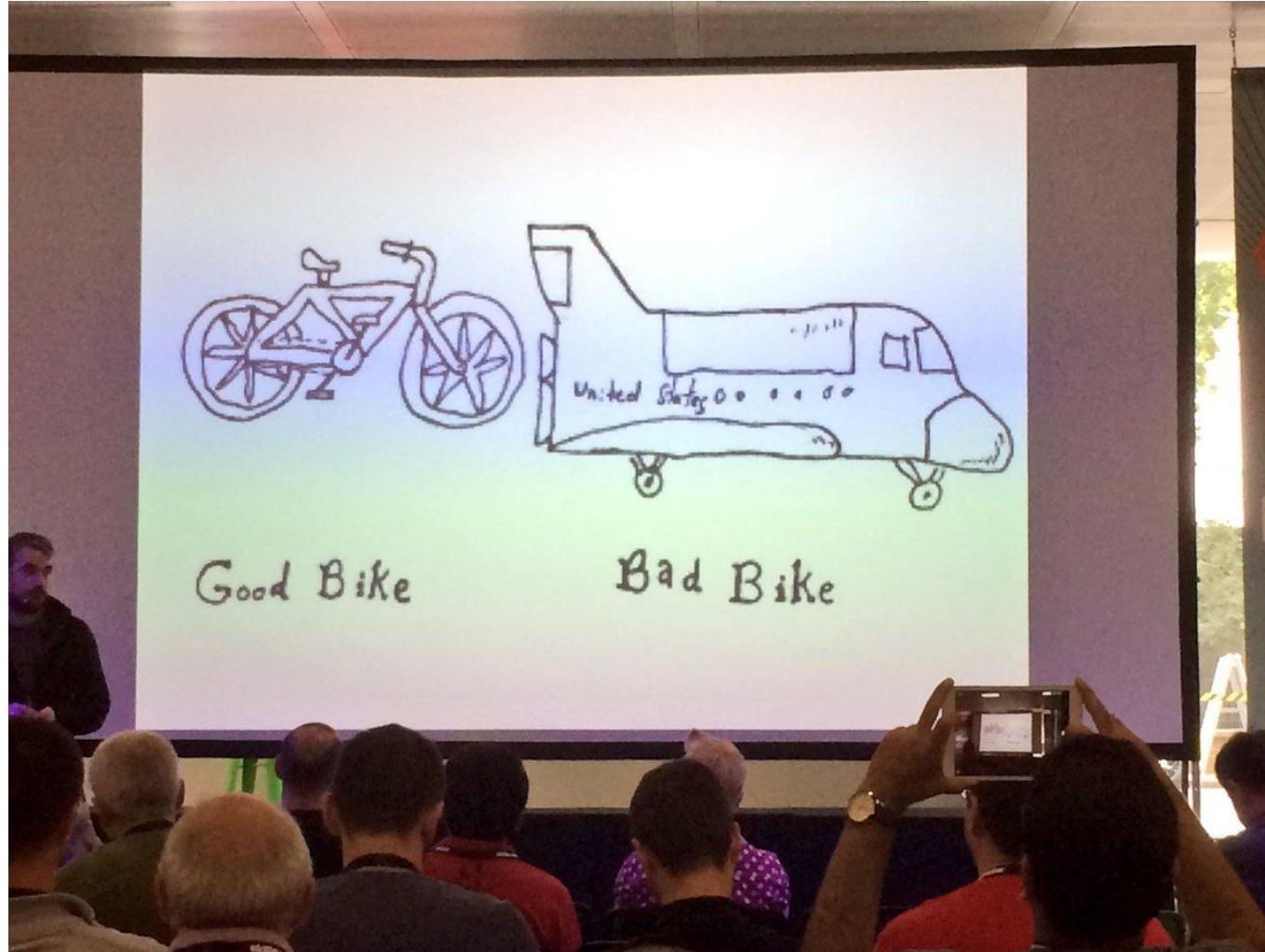
# *Un développement itératif et incrémental ... Oui, mais bien fait ... avec de la valeur métier !*



# Restez KISS, évitez l'Over-Engineering !



This! @gregyoung on Over-Engineering #dddx



# Un développement agile de qualité

Développement dirigé par les tests

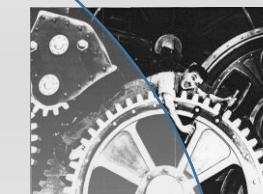
Le bon produit  
(*right Product*)

Exigences agiles :  
**User Story**  
Spécification par l'exemple



Le produit  
correctement écrit  
(*product right*)

eXtreme Programming  
Mise en avant des tests  
Excellence Technique  
mouvement Software Craftsmanship



Travail collaboratif

Le produit vite fait  
et bien fait

(livraison fréquente d'un logiciel  
opérationnel avec une forte valeur métier)

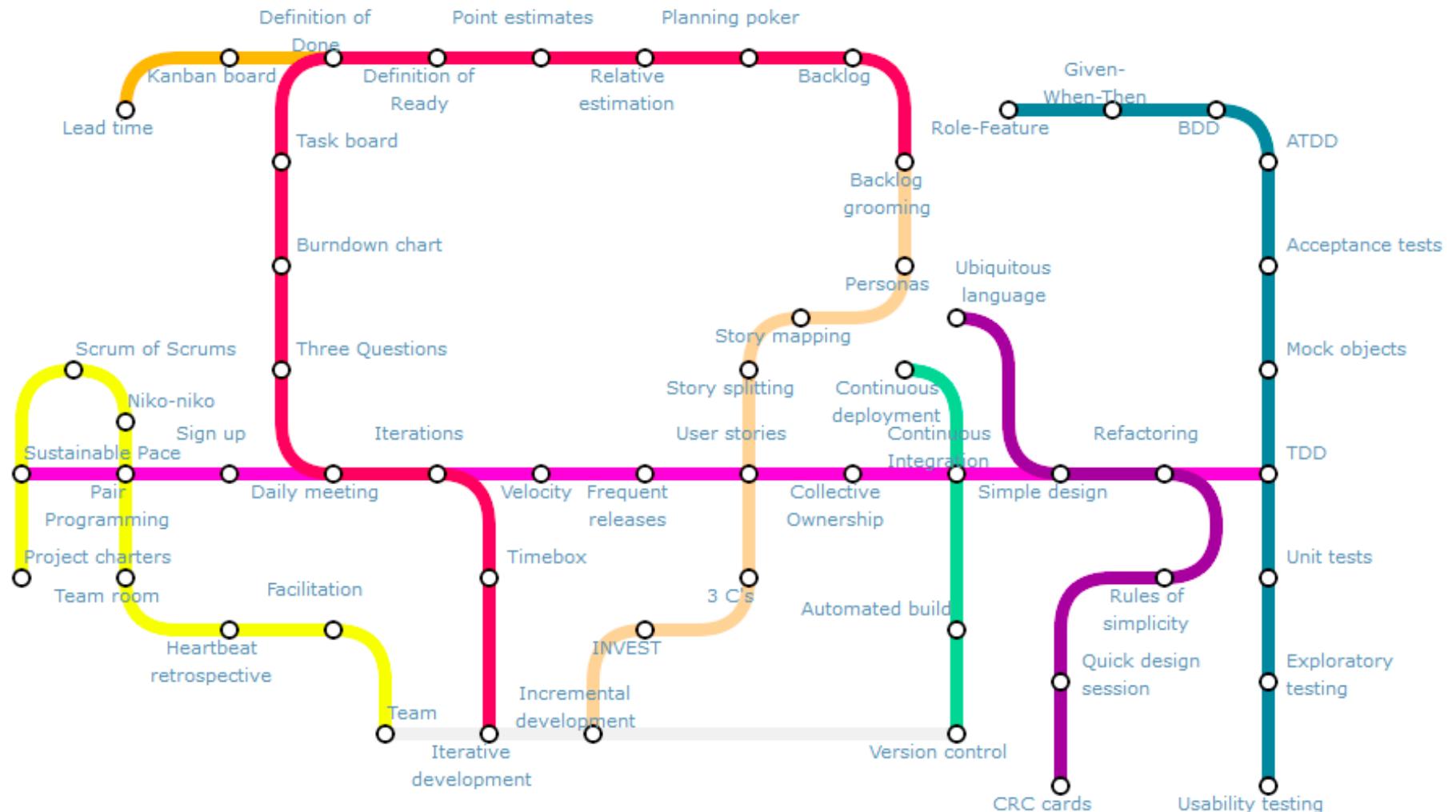
**Scrum**  
Kanban

Responsabilité  
collective

Démarche itérative incrémentale

# Ecosystème des pratiques agiles

(<http://guide.agilealliance.org/subway.html>)

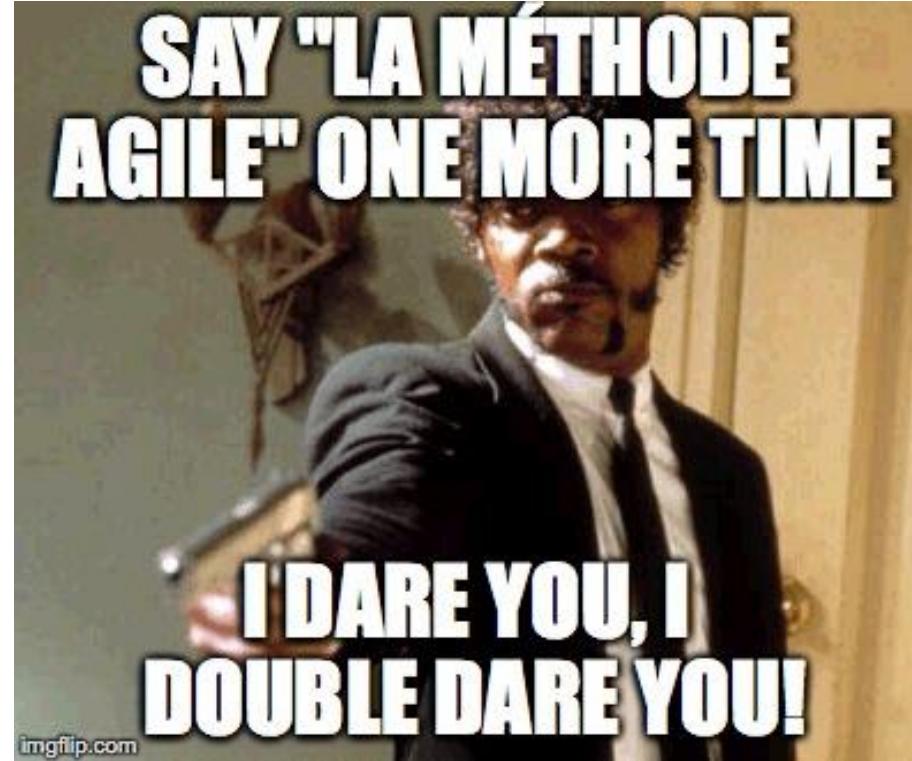


Lines represent practices from the various Agile "tribes" or areas of concern:

Extreme Programming  
Teams  
Lean

Scrum  
Product management  
Devops

Design  
Testing  
Fundamentals



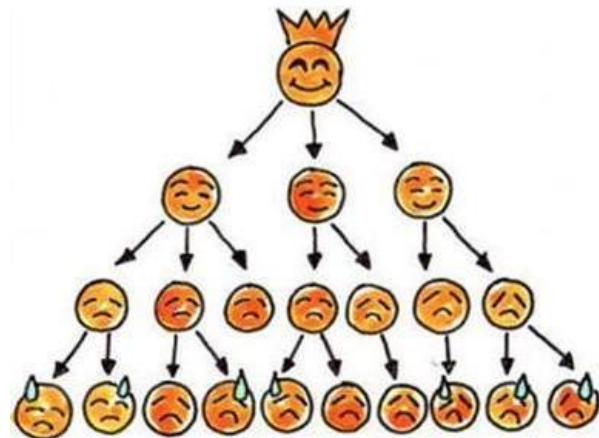
Extrait : <http://blog.soat.fr/2016/02/la-methode-agile-nexiste-pas>

Voir aussi : [https://medium.com/@cyril\\_lakech/l-agile-n-est-plus-sexy-c-est-la-norme-le-nouveau-challenge-des-agilistes-survivre-76fcf467fe6f#.kjbmrqsu](https://medium.com/@cyril_lakech/l-agile-n-est-plus-sexy-c-est-la-norme-le-nouveau-challenge-des-agilistes-survivre-76fcf467fe6f#.kjbmrqsu)

# L'agilité : un *savoir-faire* ET un *savoir être* ...

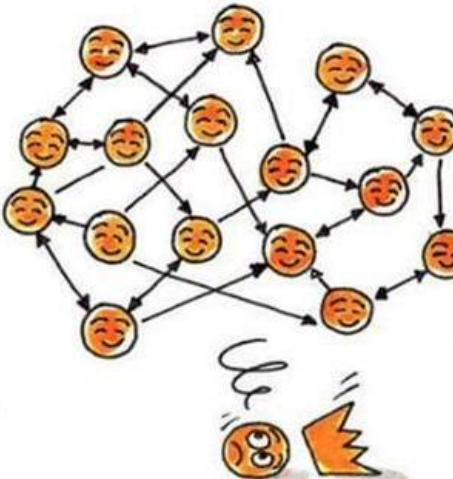
Un nouvel état d'esprit  
à la conquête d'une nouvelle culture d'entreprise ...

Depuis toujours, c'est ainsi...



<https://twitter.com/kaizenmag/status/551055626005401600>

Mais un jour peut-être !



Co-création  
Co-décision  
Collaboration  
Communication

les règles de l'**INTELLIGENCE COLLECTIVE**



ÉCOUTER avec **ATTENTION**



PARLER avec **INTENTION**



ÊTRE **BIENVEILLANT**



SE FAIRE **CONFiance**



RESPECTER le **CADRE**

CC-BY-NC-SA

<http://www.recompose.it/2014/11/26/principes-intelligence-collective/>

@heleneponville  
& www.recompose.it

# Et un jargon agile à s'approprier ...

## Scrum en 100 mots

SCRUM est une méthodologie AGILE.

Elle vise à satisfaire au mieux le besoin du PRODUCT OWNER en l'impliquant aux différentes phases du projet. Celui-ci est responsable du PRODUCT BACKLOG, liste priorisée des USERS STORIES.

Lors du SPRINT PLANNING, le PRODUCT OWNER, le SCRUM MASTER et la TEAM définissent le SPRINT BACKLOG que la TEAM s'engage à réaliser au cours du SPRINT.

Au quotidien, le SCRUM MASTER se réunit avec la TEAM pour le DAILY SCRUM et la BURN DOWN CHART est mise à jour.

La TEAM conclut le SPRINT par une DEMO du travail réalisé, puis une SPRINT REVIEW.

Extrait de : <http://blog.soat.fr/2010/05/un-apercu-de-scrum-pour-neophyte-%E2%80%A6/>

# Quelques liens pour s'y retrouver dans le jargon agile

<http://guide.agilealliance.org>

<http://referentiel.institut-agile.fr/>

<http://www.mountaingoatsoftware.com/agile>

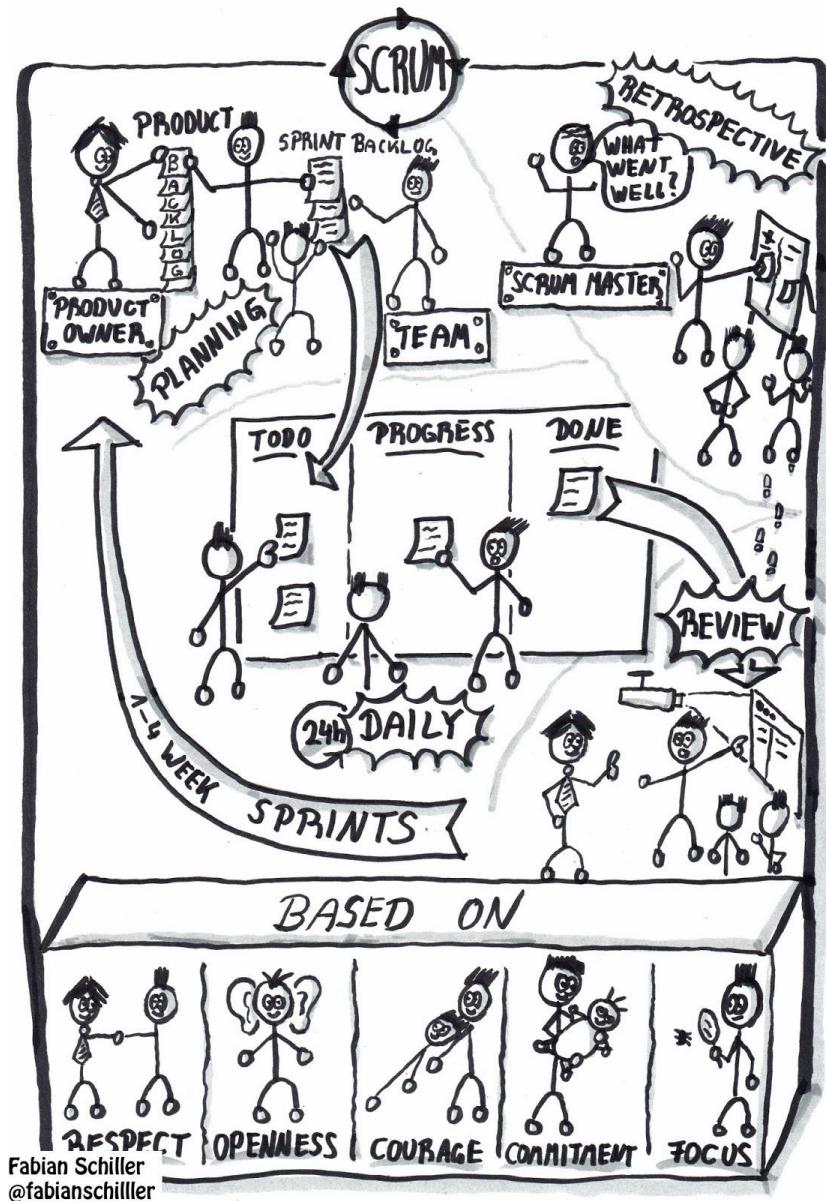
<https://www.scrum.org/Resources/Scrum-Glossary>

<http://www.aubryconseil.com/post/2007/07/17/262-glossaire-scrum>

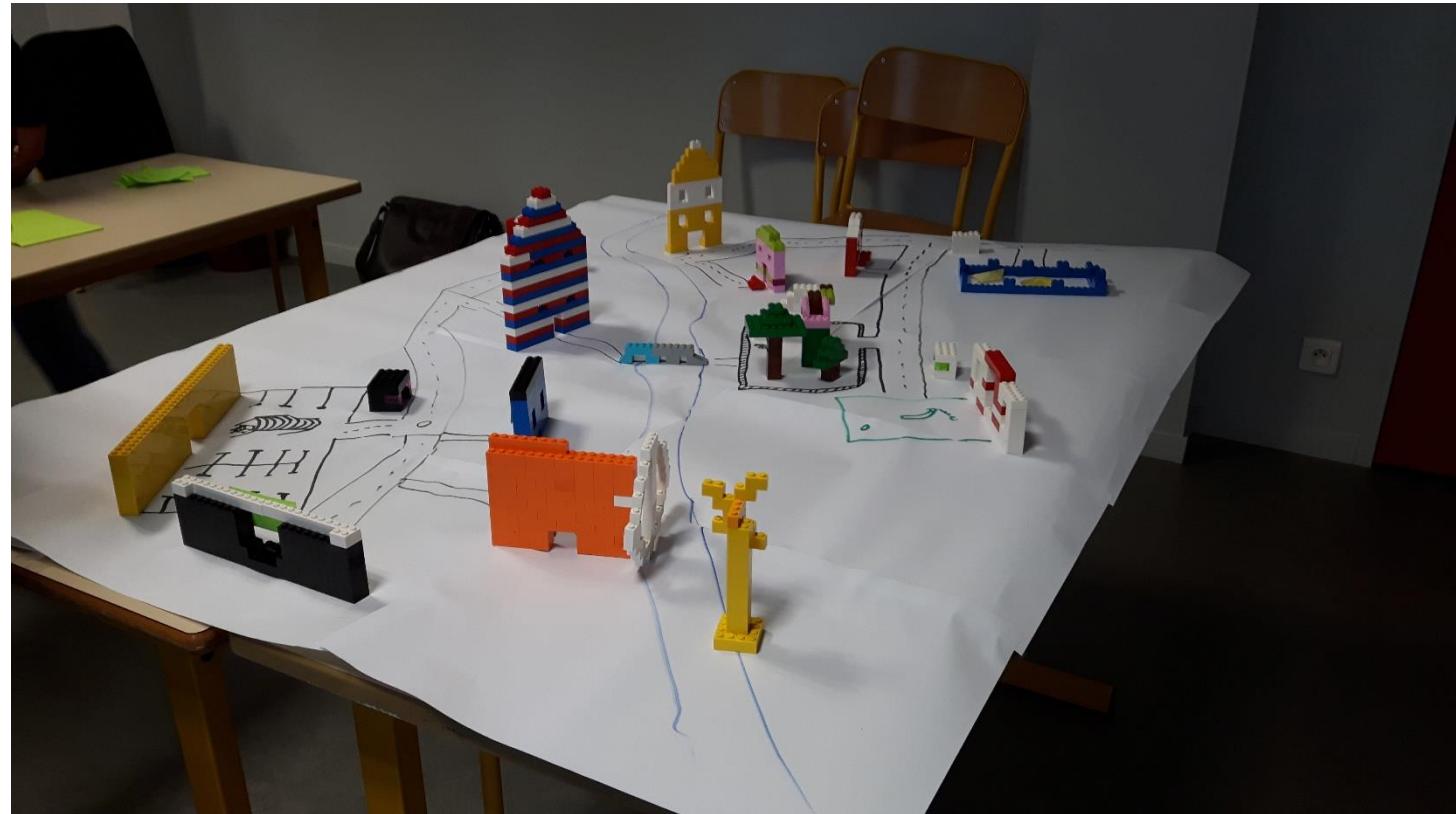
<http://thierry-leriche-dessirier.developpez.com/tutoriels/general/memento-scrum-destination-equipe>



# La semaine dernière ...

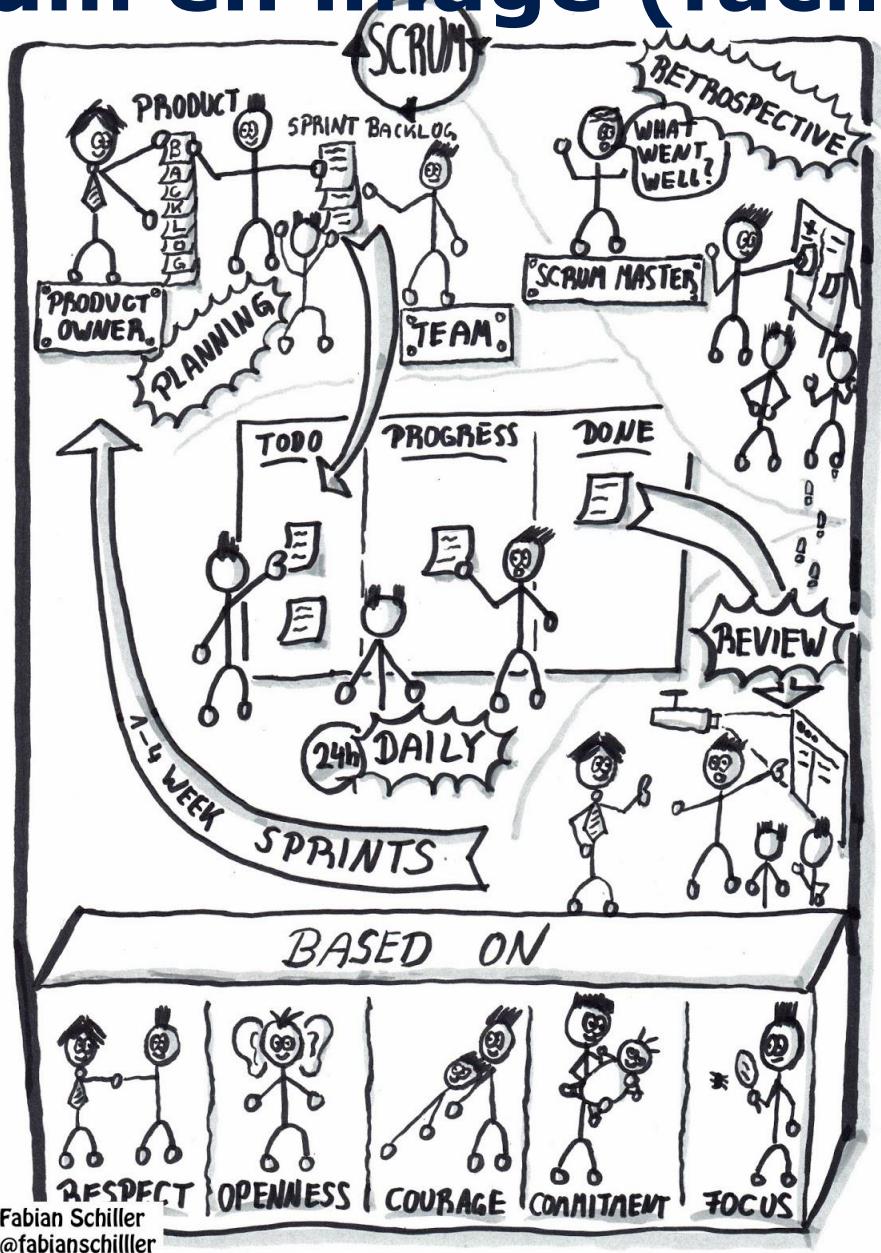


## SCRUM SIMULATION WITH LEGO



En savoir plus sur : <http://www.lego4scrum.com/>

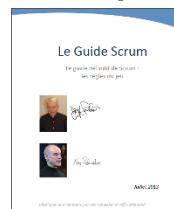
# Scrum en image (facilitation graphique)



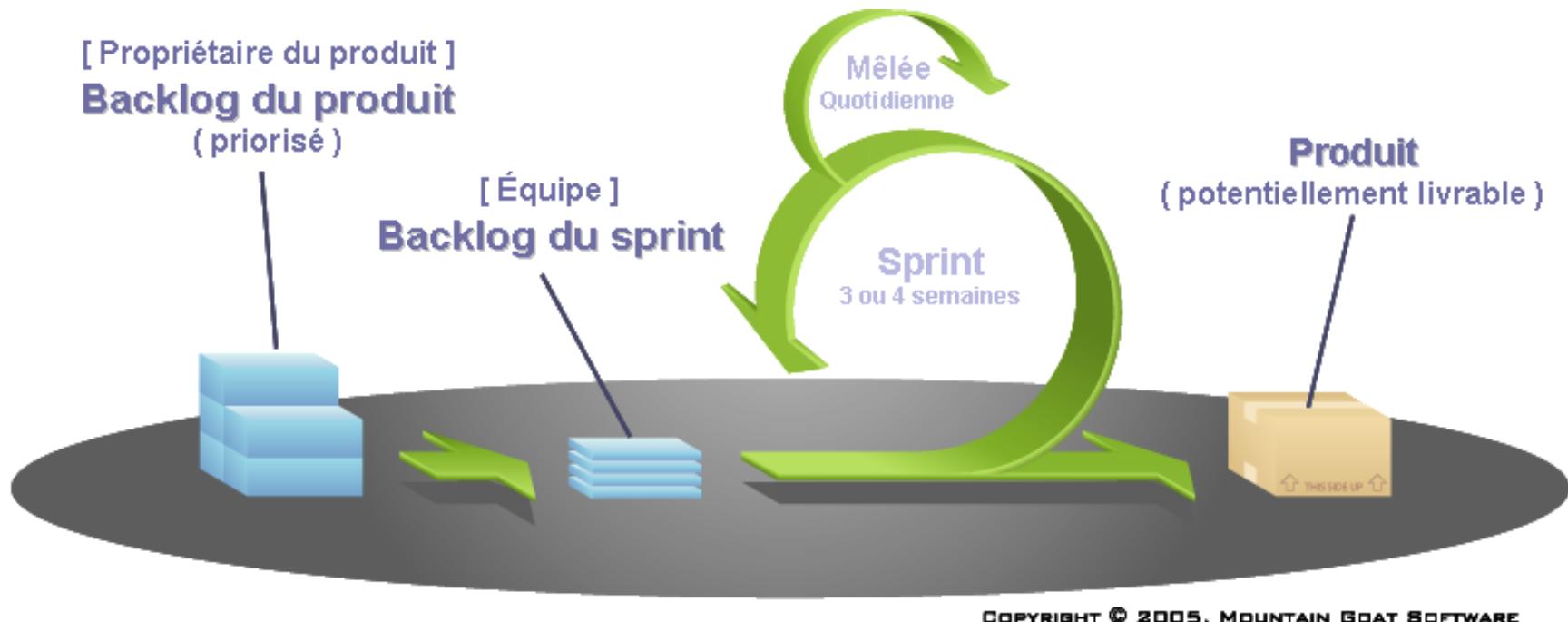
## Scrum:

Un cadre de travail permettant de répondre à des problèmes complexes et changeants, tout en livrant de manière productive et créative des produits de la plus grande valeur possible

(Définition extraite du Guide Scrum)



# Scrum : une méthodologie orientée Processus



3

## Rôles

Product Owner

Scrum Master

Equipe de développement

3

## Cérémonies

Sprint Planning

Daily Scrum (Mélée quotidienne)

Sprint Review et retrospective

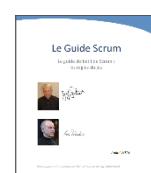
3

## Artefacts

Product Backlog

Burndown Chart

Sprint BackLog



**3 Cérémonies**

**Sprint Planning**

**Daily Scrum**

**Sprint Review**

# La planification de sprint ...

3 Cérémonies  
Sprint Planning  
Daily Scrum  
Sprint Review

L'équipe s'engage à réaliser les stories choisies.

**Objectif du Sprint & choix des Stories** qui seront traitées dans ce sprint

\* Teasing \* ☺  
**Découpage** de chaque story en « **tâches** »

Découpage du travail à faire

\* Teasing \* ☺  
**Estimation de chaque tâche**

peut-être #noestimate si story de très faible granularité  
le but de cette étape est d'affiner les derniers détails

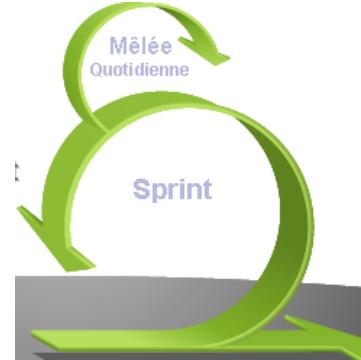
**Mise en place du « *task board* » et répartition des premières tâches**

# La mélée quotidienne pour s'assurer du bon déroulement du sprint

3 Cérémonies  
Sprint Planning  
**Daily Scrum**  
Sprint Review

## Point quotidien de 15 minutes

Chacun répond au trois 3 questions suivantes :



*Qu'est ce que j'ai fait hier ?*

*Qu'est ce que je prévois de faire aujourd'hui ?*

*Qu'est-ce qui me bloque, quels obstacles ai-je rencontré ?*

Ces réunions se passent habituellement debout ce qui leur vaut également le nom de « **stand-up** » meetings.

# La revue de Sprint



Où en est mon produit ?

**3 Cérémonies**  
Sprint Planning  
Daily Scrum  
**Sprint Review**

**Démo** : présentation au travers d'un scénario des stories réalisées et validées au cours du sprint (considérées comme finies)

**Mise à jour du backlog** : retour dans le backlog des éléments non terminés

**Calcul de la vélocité pour chaque équipe (stories finies et validées)**

**Mise à jour du burndown chart**

**Rétrospective sur le sprint**

# Quelques mots sur la vélocité ...

## Vélocité

**Volume de travail réalisé au cours d'un sprint**

***Addition des estimations des stories réalisées au cours du sprint et acceptées lors de la démo***

### Remarques :

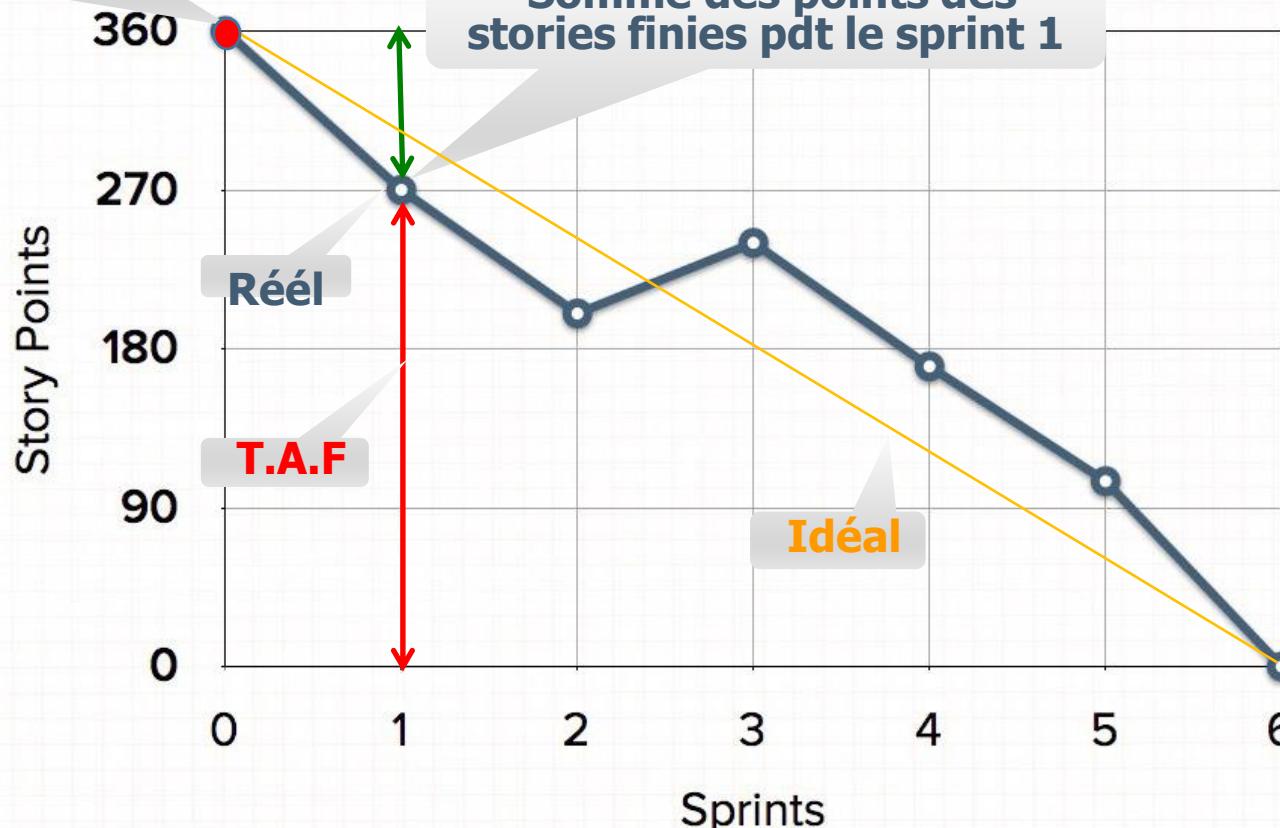
- La Vélocité ne tient compte que des stories terminées
- La Vélocité mesure un volume d'efforts et non un résultat économique

# ... et le BurndownChart de release

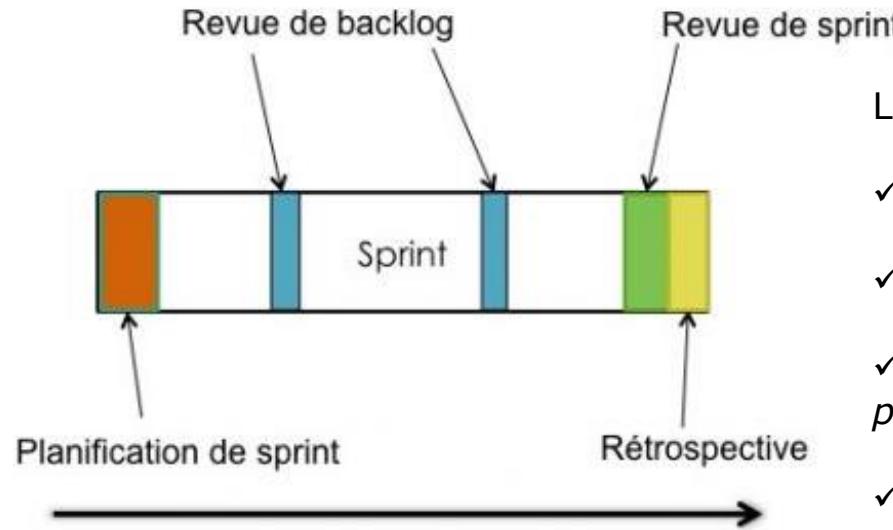
**Le Burndown Chart** est une représentation graphique du Travail A Faire (TAF), actualisé à la fin du sprint .

Somme des estimations en points de toutes les stories du backlog

Somme des points des stories finies pdt le sprint 1



# Pour information « dans la vraie vie » : un exemple de fréquence des réunions d'un sprint de 3 semaines



Les durées, à titre indicatif, pour un sprint de 3 semaines :

- ✓ **Réunion de planification de sprint** : 2h
- ✓ **Scrum quotidien** : 1/4h
- ✓ **Revue de backlog** : 1h (deux fois)  
*permet d'avancer collectivement sur le bac(log) de culture.*
- ✓ **Revue de sprint** : 1h15
- ✓ **Rétrospective** : 1h15

## **3 Artefacts**

Product Backlog

Burndown Chart

Sprint BackLog

# Un management Visuel pour une transparence des artefacts



Product Backlog dans un Story Mapping par exemple

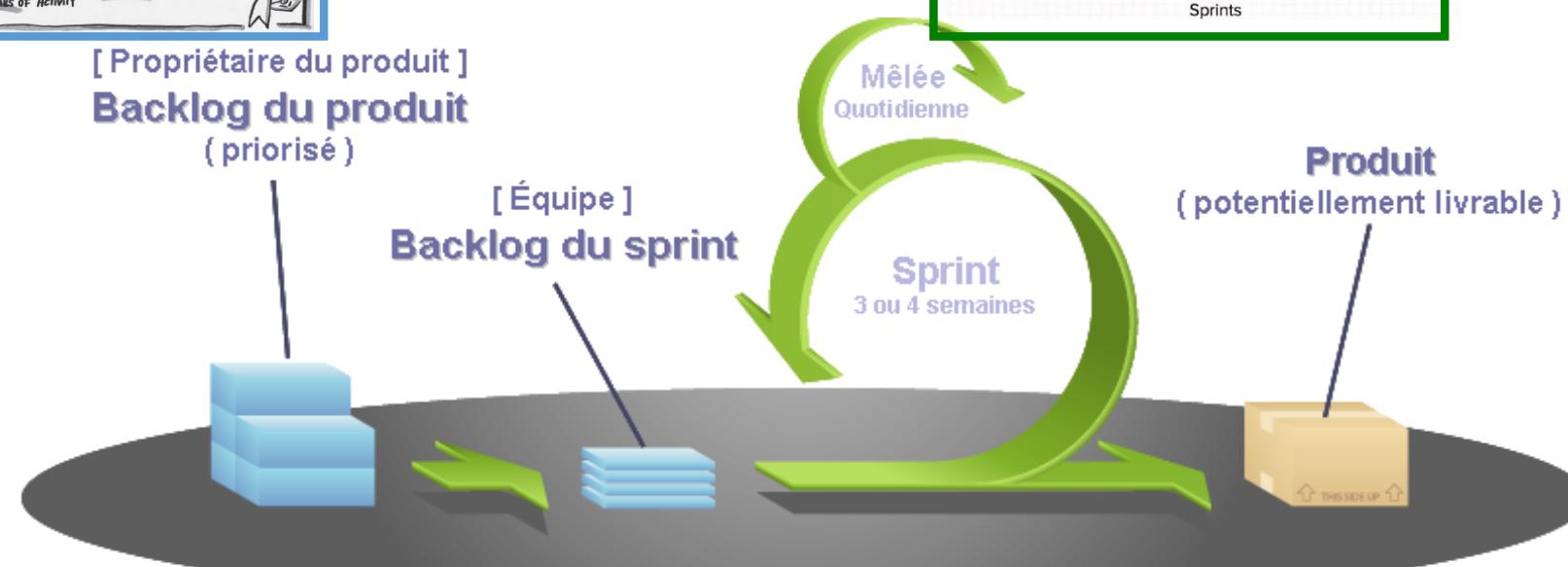


Burndown Chart

[ Propriétaire du produit ]  
Backlog du produit  
(priorisé)

[ Équipe ]  
Backlog du sprint

Produit  
(potentiellement livrable)



COPYRIGHT © 2005. MOUNTAIN GOAT SOFTWARE

Sprint Backlog & Task Board

Story	To Do	In Process	To Verify	Done
As a user, I... 8 points	Code the... 9 Test the... 8	Code the... 4 Test the... 8	Test the... SC 6	Code the... 5 Test the... 4 Test the... SC 6 Test the... SC 6
	Code the... 2 Test the... 8	Code the... 8 Test the... 4		
	Test the... 8 Test the... 4			
As a user, I... 5 points	Code the... 8 Code the... 4	Test the... 8 Code the... 6	Code the... 8	Test the... SC 6 Test the... SC 6 Test the... SC 6

# Transparence des artefacts

Scrum repose sur la **transparence**.

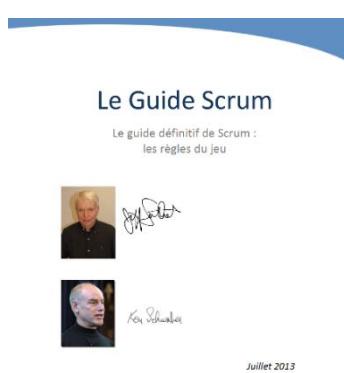
Les décisions pour **optimiser la valeur** et **contrôler le risque** sont prises en se basant sur l'état perçu des artéfacts.



Dans la mesure où la transparence est complète, ces décisions ont une base solide.

Dans la mesure où les artéfacts ne sont pas totalement transparents, ces décisions peuvent être faussées, la valeur moindre et le risque accru.

Extrait de : <http://www.scrumguides.org>



# Zoom sur l'indispensable Definition of Done : Un 4ème artefact ?

## Définition de 'fini'

Pratique

### De quoi s'agit-il?

L'équipe affiche de façon visible une liste de critères génériques qui conditionnent le fait de pouvoir considérer un incrément comme "fini". Faute de remplir ces critères en fin de Sprint ou d'itération le travail réalisé n'est pas comptabilisé dans la vélocité.

### Comment reconnaître son utilisation?

- l'équipe est capable de montrer, sur demande, sa définition de "fini"
- ces critères sont évoqués en fin de Sprint ou d'itération et justifient la décision de comptabiliser un incrément dans la vélocité, ou non

### Origines

- Proposée par Dan Rawsthorne en 2002-2004
- Répandue sous ce terme à partir de 2007 environ
- Considérée comme un élément majeur de Scrum

# Exemple d'une Définition of Done



**Peetic**

## Définition de « fini »

Fonctionne conformément aux discussions et à la user stories

La validation de la « user story » est faite sur la plate-forme d'intégration continue

Toutes les tâches sont passées par une revue d'un autre membre de l'équipe ou réalisées en *pair programming*

Toutes les « user stories » ayant une interface web sont testées avant validation sur Firefox, Chrome et IE.

Les tests d'acceptance sont transformés en tests automatisés fonctionnels

Des tests unitaires sont réalisés sur chaque « user story » (pas de % de couverture) ou sur chaque bug fermé.

Rien d'inutile n'est présent

## **3 Rôles**

Product Owner  
Scrum Master  
Equipe de développement

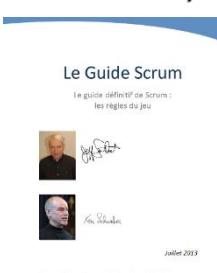
# Le Product Owner d'après le Guide Scrum

Le Product Owner est responsable de maximiser la valeur du produit et du travail de l'Équipe de Développement. La façon de jouer ce rôle peut varier grandement selon les entreprises, les Équipes Scrum et les individus.

Le Product Owner est la seule personne responsable de gérer le carnet de produit (*Product Backlog*). La gestion du Product Backlog comprend :

- Exprimer clairement les items du Product Backlog ;
- Ordonner les items du Product Backlog pour mieux réaliser les objectifs et missions ;
- Optimiser la valeur du travail effectué par l'Équipe de Développement ;
- S'assurer que le Product Backlog est visible, transparent, et clair pour tous, et qu'il montre ce sur quoi l'Équipe de Développement travaillera prochainement ; et,
- S'assurer que l'Équipe de Développement comprend adéquatement les items du Product Backlog.

Le Product Owner peut lui-même accomplir les tâches susmentionnées ou les déléguer à l'Équipe de Développement. Toutefois, le Product Owner demeure responsable de ces dernières.



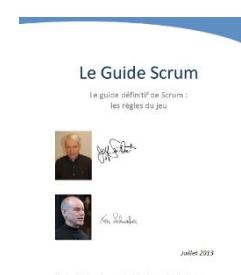
En savoir un peu plus sur les activités du PO : voir annexe

Isabelle BLASQUEZ - 2016

# Le Scrum Master

Le Scrum Master est responsable de s'assurer que Scrum est compris et mis en œuvre. Les Scrum Masters remplissent leur rôle en s'assurant que l'Équipe Scrum adhère à la théorie, aux pratiques et aux règles de Scrum.

Le Scrum Master est un leader au service de l'Équipe Scrum. Le Scrum Master aide ceux qui sont externes à l'Équipe Scrum à comprendre lesquelles de leurs interactions avec l'Équipe Scrum sont bénéfiques et lesquelles ne le sont pas. Le Scrum Master aide tout le monde à changer ces interactions pour maximiser la valeur créée par l'Équipe Scrum.



En savoir un peu plus ... : voir annexe

Isabelle BLASQUEZ - 2016

# L'équipe de développement

L'Équipe de Développement est constituée de professionnels qui livrent à chaque Sprint un incrément « terminé » et potentiellement livrable du produit. Seuls les membres de l'Équipe de Développement créent l'incrément.

L'Équipe de Développement possède les caractéristiques suivantes :

- Elle est **auto-organisée**. Nul (même pas le Scrum Master) n'indique à l'Équipe de Développement comment transformer les items du Product Backlog en incréments de fonctionnalités potentiellement livrables.
- Elle est **pluridisciplinaire** avec toutes les compétences nécessaires pour créer un incrément du produit ;
- Scrum ne reconnaît **aucun titre aux membres de l'Équipe de Développement** autre que celui de développeur, indépendamment du travail effectué par cette personne ; il n'y a pas d'exception à cette règle ;
- Scrum ne reconnaît **pas d'équipes** à l'intérieur de l'Équipe de Développement indépendamment des domaines spécifiques qui doivent être couverts tels que l'exécution de tests ou l'analyse fonctionnelle ; il n'y a pas d'exception à cette règle ; et,
- Les membres de l'Équipe de Développement peuvent détenir individuellement des compétences et des centres d'intérêt spécifiques, **mais c'est l'Équipe de Développement dans son ensemble qui est tenue responsable**.



Le Guide Scrum  
Le guide définitif de Scrum :  
les règles du jeu

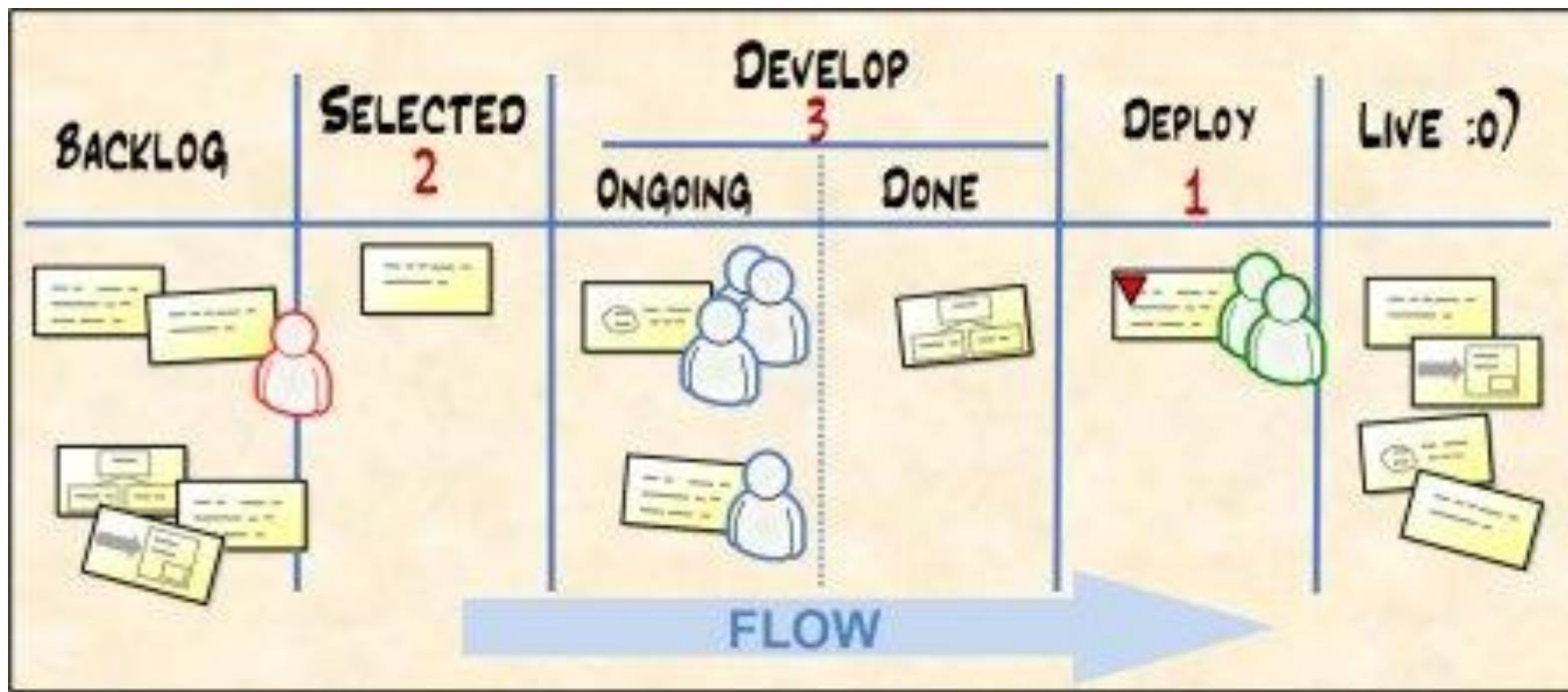


juillet 2013

En savoir un peu plus ... : voir annexe

Isabelle BLASQUEZ - 2016

# Kanban : organiser l'ensemble pour améliorer le processus de développement



Kanban est basé sur un modèle de **flux tiré** qui encourage un engagement au plus tard, à la fois sur la priorisation du travail et sur la livraison du travail en cours

## 5 Pratiques

Visualiser le flux des travaux

Limiter le nombre de travaux en cours

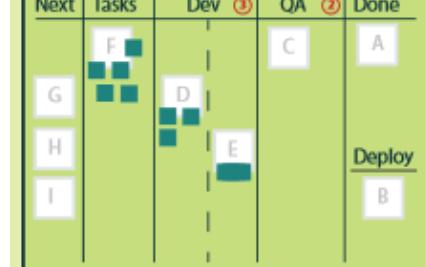
Gérer le flux

Rendre le processus explicite

Identifier des opportunités d'améliorations

Developed at Toyota in the 1940's, designed as a pull-based, self-stocking system for automobile production. Adopted as a software development management technique in 2003, focusing on "work in progress" limitation to achieve short cycle times (= time from start to finish). The 5 steps:

### Visualize the Workflow



### Manage Flow

Is cycle time going up or down? Where are bottlenecks?  
Is something stuck somewhere, or should tasks be merged/split up?

### Explicit Policies

Identify and define policies for standard tasks, tasks with fixed delivery dates, intangible tasks, etc. Make sure that even exceptional tasks are integrated well into the process.



### 1

Identify the stages in your project. Capture them on a board. Each stage is a column. Cards represent the tasks that have to go through in process.

### 2

Limit the maximum workload on specific stages. Because tasks are pulled from column to column not pushed, each stage can only work on as many tasks as the next stages allow. This way you won't overload your team members and can identify bottlenecks.

### 3

Talk about what worked and what didn't. Identify bottlenecks and fix them. Kanban is evolutionary and lives of the experiments you do with WIPs, policies, etc. Run the experiments using a scientific method.

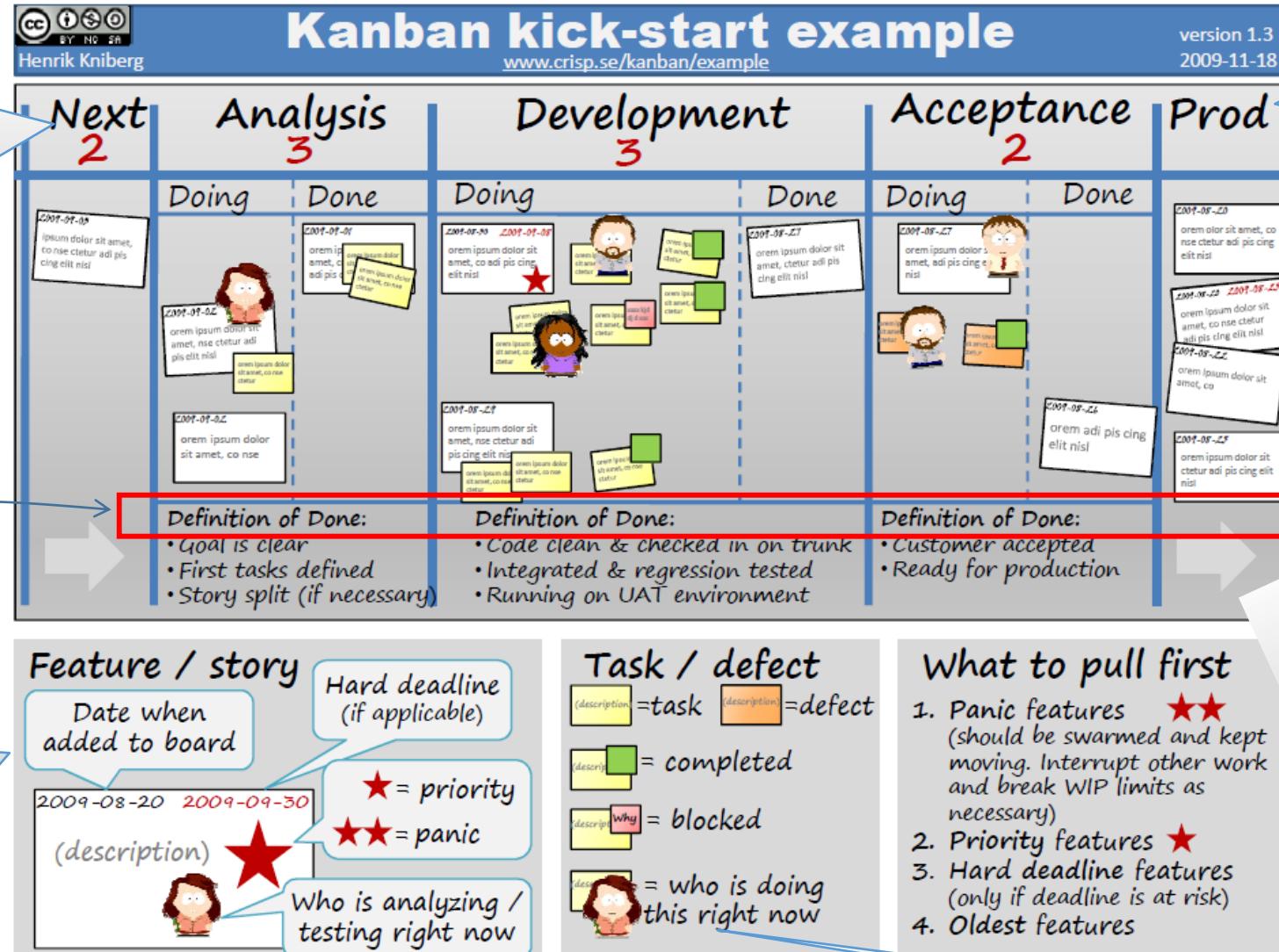


### 4

### Explicit Policies

Identify and define policies for standard tasks, tasks with fixed delivery dates, intangible tasks, etc. Make sure that even exceptional tasks are integrated well into the process.

# « Kanbaniser » et personnaliser votre radiateur d'informations



Ajouter des colonnes par rapport à un simple tableau des tâches

En fonction de vos besoins

Adopter un code couleur

Isabelle BLASQUEZ - 2016

Extrait : <http://www.crisp.se/file-uploads/kanban-example.pdf>

A voir aussi : <http://toolsforagile.com/blog/archives/1045/infothographic-ingredrients-of-kanban>

<http://www.infoq.com/minibooks/kanban-scrum-minibook>

# XP : une méthodologie proposant aussi des pratiques techniques

5

Valeurs

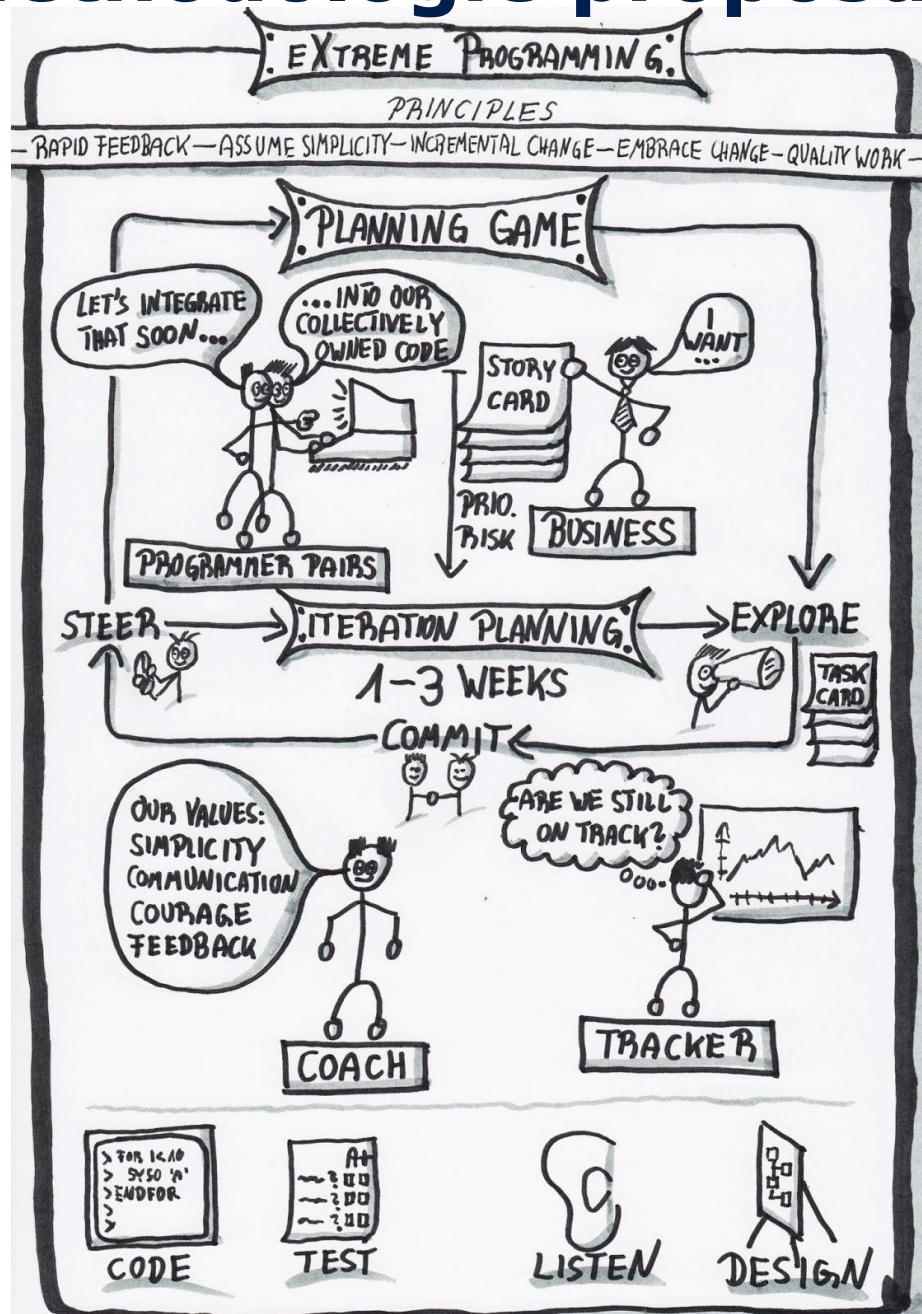
Communication

Simplicité

Feedback

Courage

Respect



13

Pratiques

Client sur site

Jeu du Planning ou Planning poker

Intégration continue

Petites livraisons

Rythme soutenable

Tests de recette (ou tests fonctionnels)

Tests unitaires

Conception simple

Utilisation de métaphores

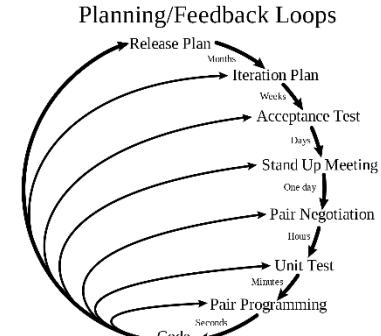
Refactoring (ou remaniement du code)

Appropriation collective du code

Convention de nommage

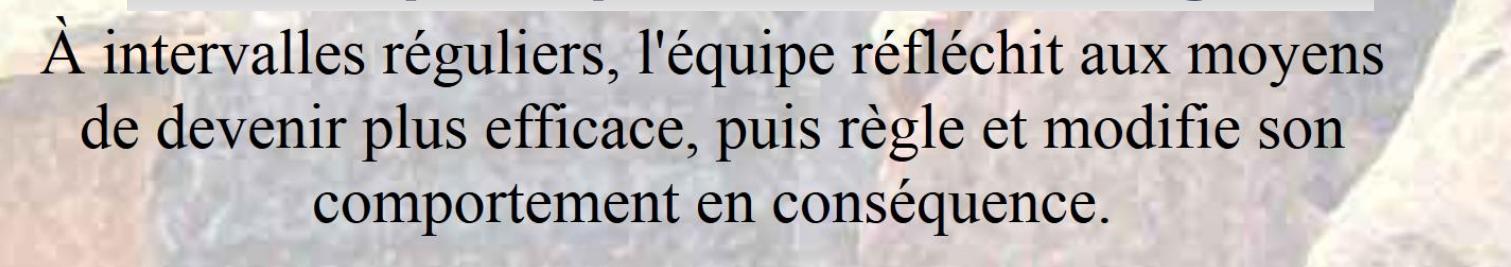
Programmation en binôme

Isabelle BLASQUEZ - 2016



# L'amélioration continue (**Kaizen**) : au cœur de l'agilité

## 12ème principe du manifeste agile



À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

- Scrum préconise la **rétrospective**,
- le Lean veille à la **suppression du gaspillage**,
- Kanban se marie très bien à la **théorie des contraintes**, etc.
- Extreme Programming met en avant les **boucles de feedback**
- ...

*L'amélioration en agilité revêt plusieurs formes, les petites avancées typiques du Kaizen constituent un puissant levier de transformation, que ce soit aux niveaux personnel, équipe, organisation.*



# Le développement agile en 2015 ...

## Top 5 Agile Techniques



## Percent of 100

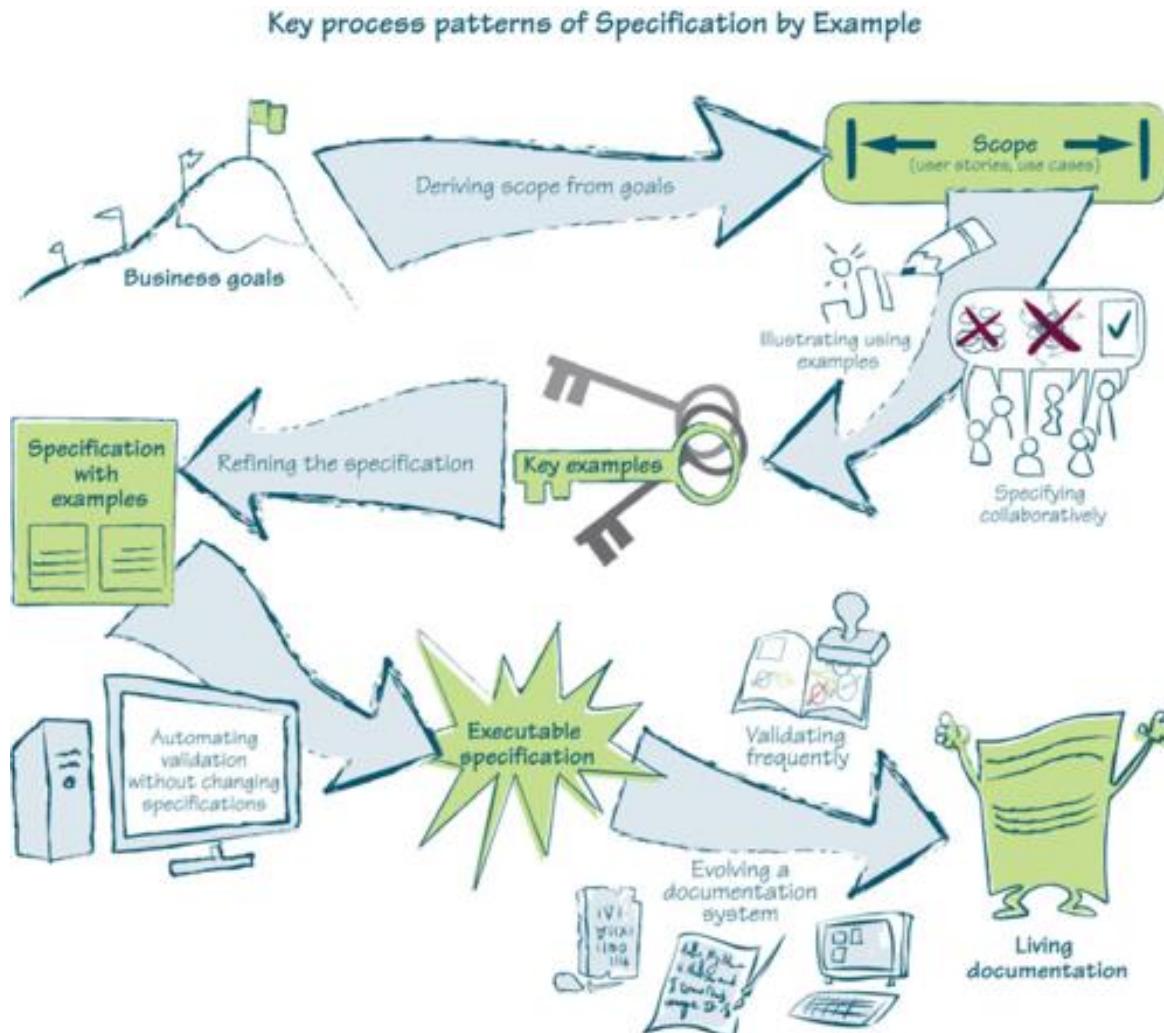
- | Technique                              | Percentage |
|--|------------|
| Daily standup                          | 80%        |
| Short iterations                       | 79%        |
| Prioritized backlogs                   | 79%        |
| Iteration planning                     | 71%        |
| Retrospectives                         | 69%        |
| Release planning                       | 65%        |
| Unit testing                           | 65%        |
| Team-based estimation                  | 56%        |
| Iteration reviews                      | 53%        |
| Taskboard                              | 53%        |
| Continuous integration                 | 50%        |
| Dedicated product owner                | 48%        |
| Single team (integrated dev & testing) | 46%        |
| Coding standards                       | 43%        |
| Open work area                         | 38%        |
| Refactoring                            | 36%        |
| Test-Driven Development (TDD)          | 34%        |
| Kanban                                 | 31%        |
| Story mapping                          | 29%        |
| Collective code ownership              | 27%        |
| Automated acceptance testing           | 24%        |
| Continuous deployment                  | 24%        |
| Pair programming                       | 21%        |
| Agile games                            | 13%        |
| Behavior-Driven Development (BDD)      | 9%         |

\*Respondents were able to make multiple selections.

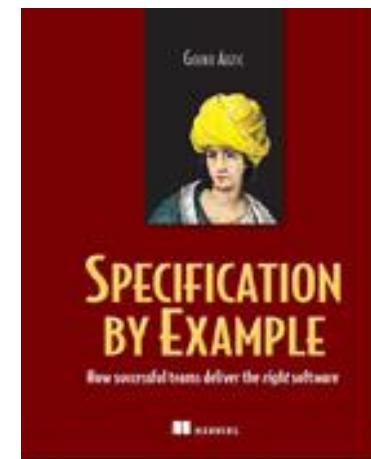
# Dans le module M3301 ...



# Une démarche *agile* de spécification par l'exemple ...



Cheminement de l'**objectif** (métier) à une **documentation vivante** au travers d'un ensemble de 7 patterns, qui permet de s'assurer que **le « bon » produit (right product)** sera effectivement livré.

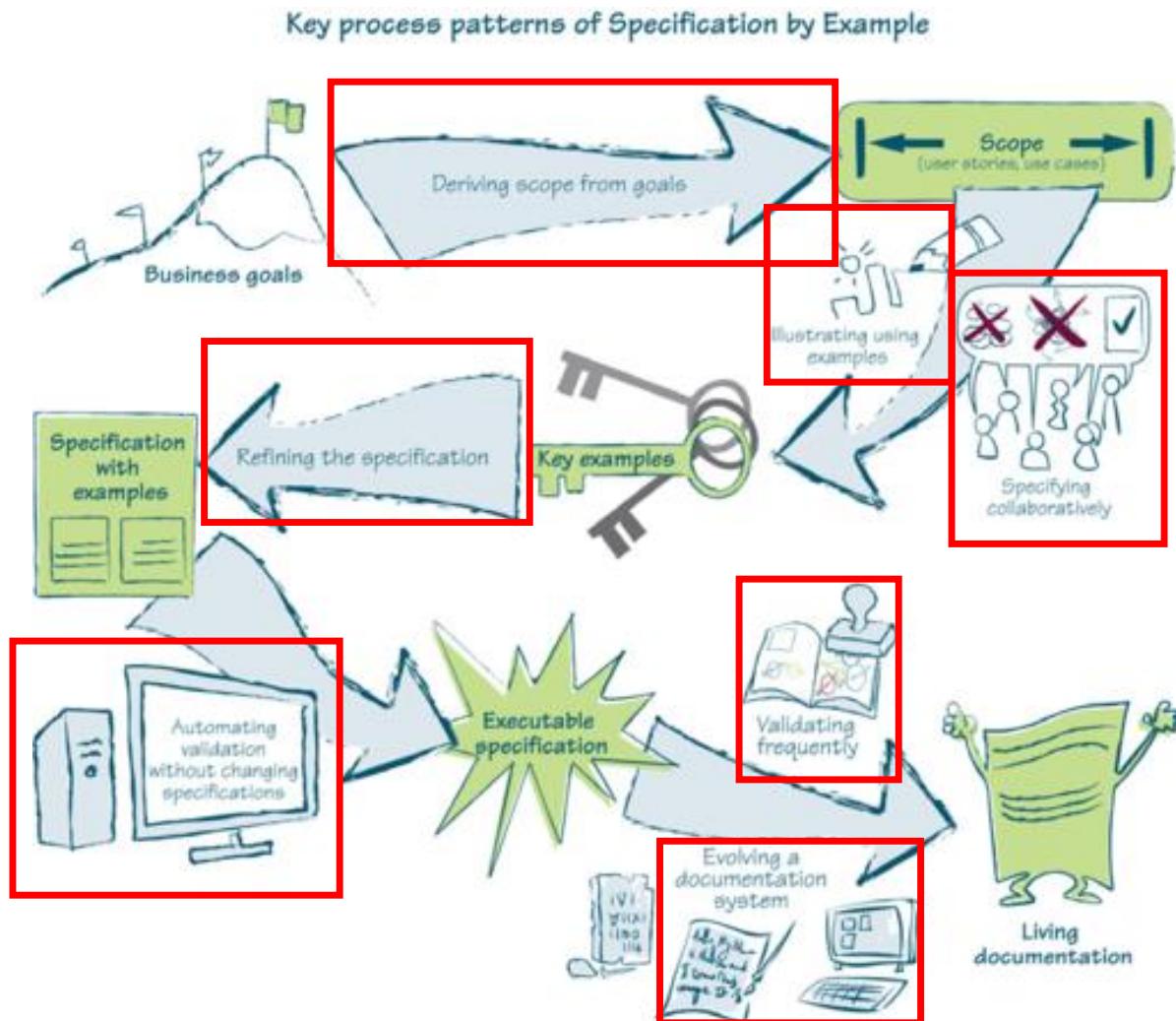


**Specification by example (SBE)** is a collaborative approach to defining **requirements** and **business-oriented functional tests** for software products based on capturing and illustrating requirements using **realistic examples** instead of abstract statements.

Extrait : [https://en.wikipedia.org/wiki/Specification\\_by\\_example](https://en.wikipedia.org/wiki/Specification_by_example)

Isabelle BLASQUEZ - 2016

# Une démarche agile de spécification par l'exemple ...



**Deriving scope from goals**  
Travail autour de la vision

## Specifying collaboratively

Compréhension commune. La Collaboration permet aux équipes de produire des spécifications qui sont faciles à comprendre.

## Illustrating requirements using example

Exemples précis, complets, réalistes

## Refining specifications

Création d'un contexte concret et précis pour le développement et le test

## Automating validation without changing specification

Une **Spécification avec des exemples** automatisés (tests) qui est compréhensible et accessible par tous les membres de l'équipe devient une **spécification exécutable**

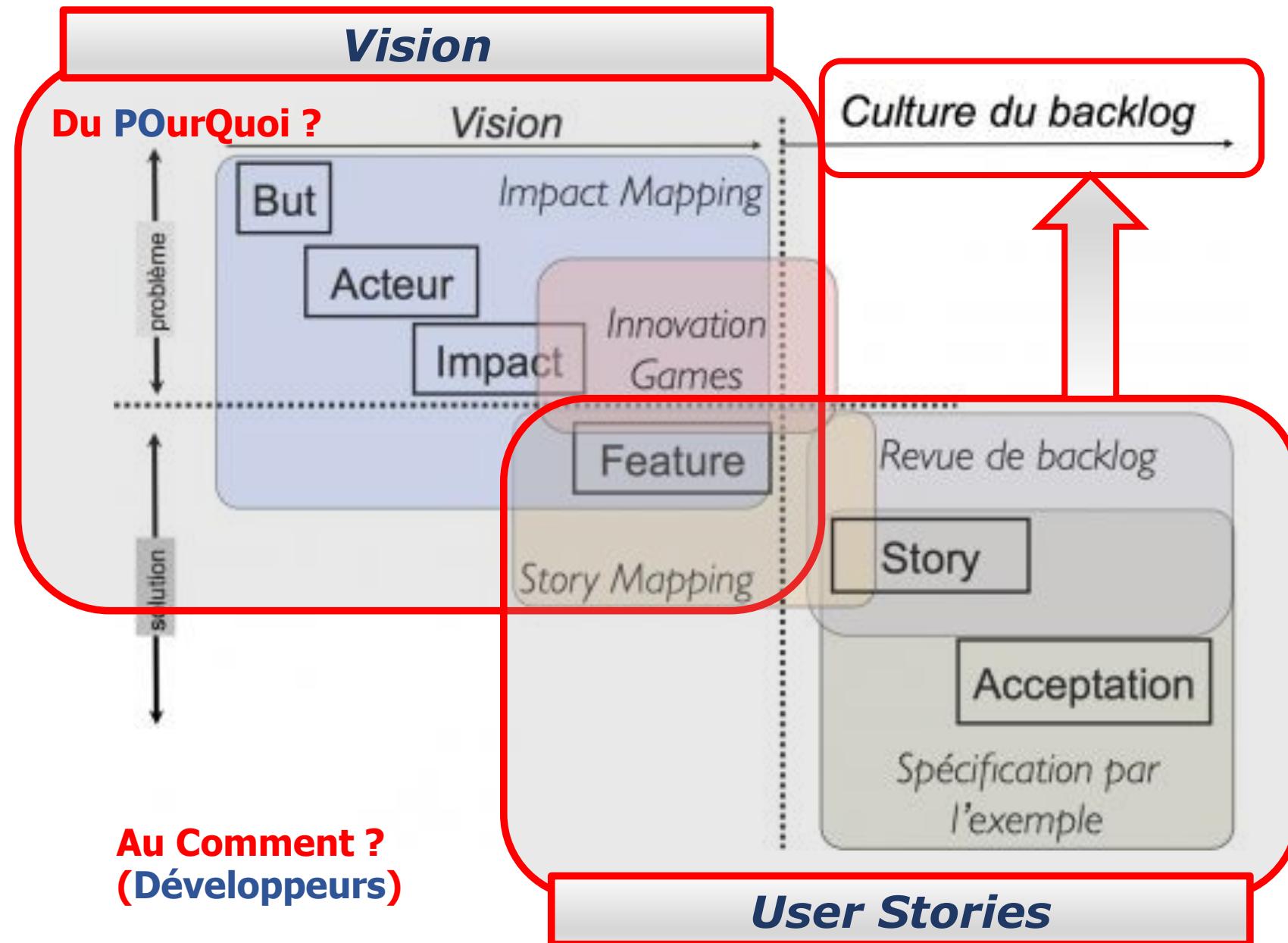
## Validating frequently

## Evolving a documentation system

Proposer une **documentation vivante** : facile à comprendre, cohérente et organisée

Extrait : [https://en.wikipedia.org/wiki/Specification\\_by\\_example](https://en.wikipedia.org/wiki/Specification_by_example)

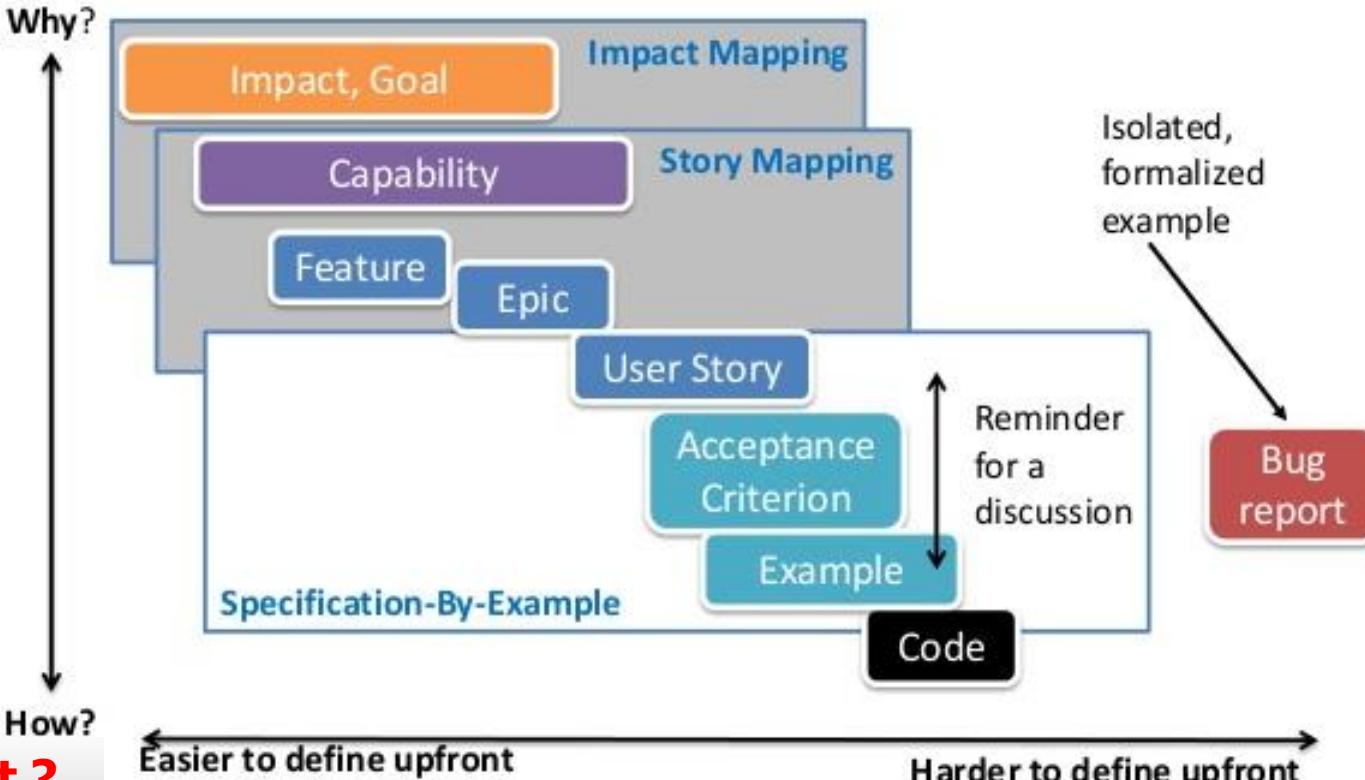
# ... allant de la Vision à l'écriture des User Stories (1/2)



# ... allant de la Vision à l'écriture des User Stories (2/2)

## Establishing a shared understanding

Du PourQuoi ?



Au Comment ?  
(Développeurs)

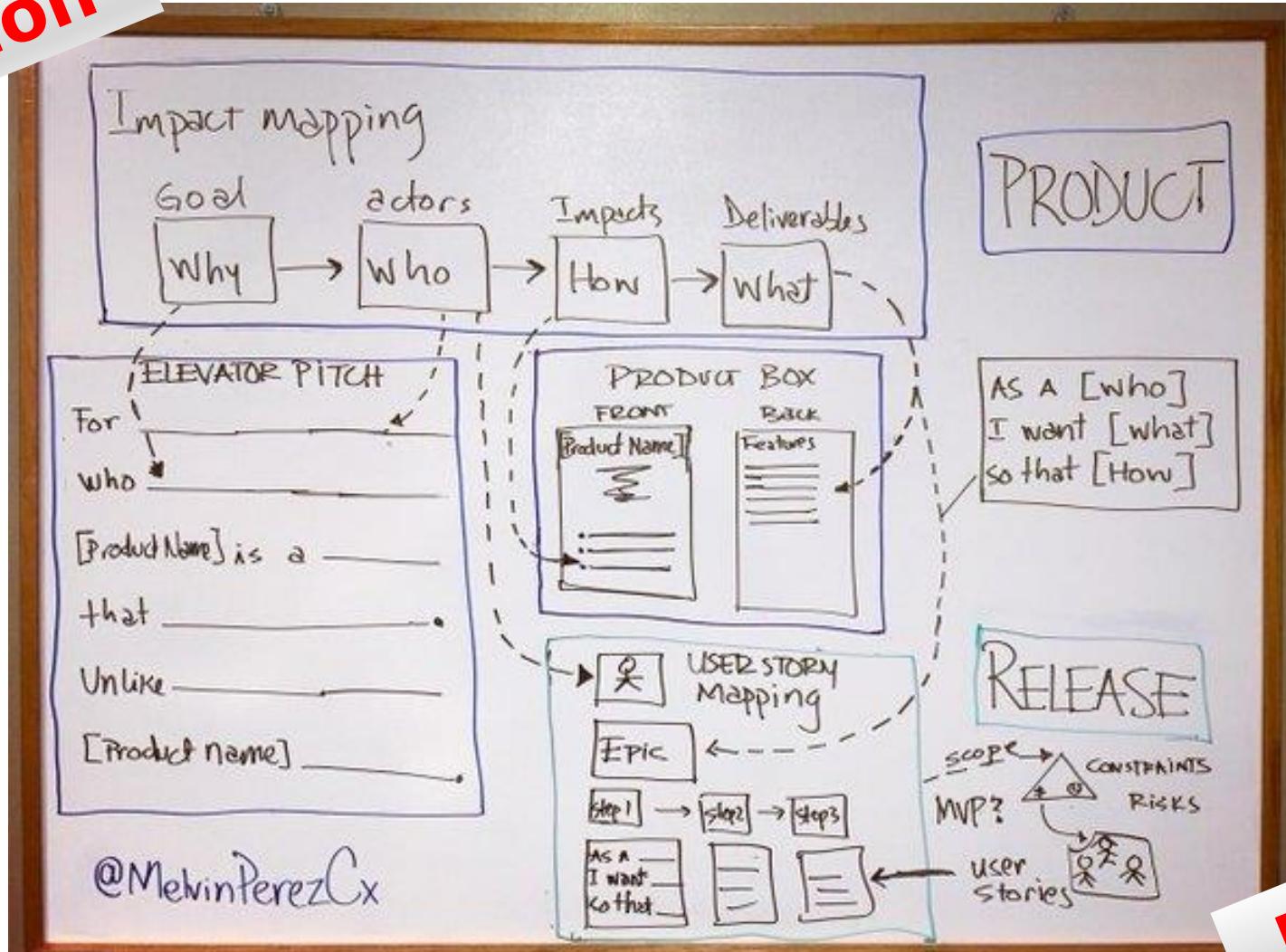


Extrait de « Specification-By-Example with Gherkin »

<http://fr.slideshare.net/chassa/2013-0603specification-byexamplewithgherkinchristianhassa>

# ... et axée autour des outils du Product Owner (right product)

Vision



User Stories

# **Annexes**

# Réflexion de l'été 2015 sur le(s) mouvement(s) agile(s)

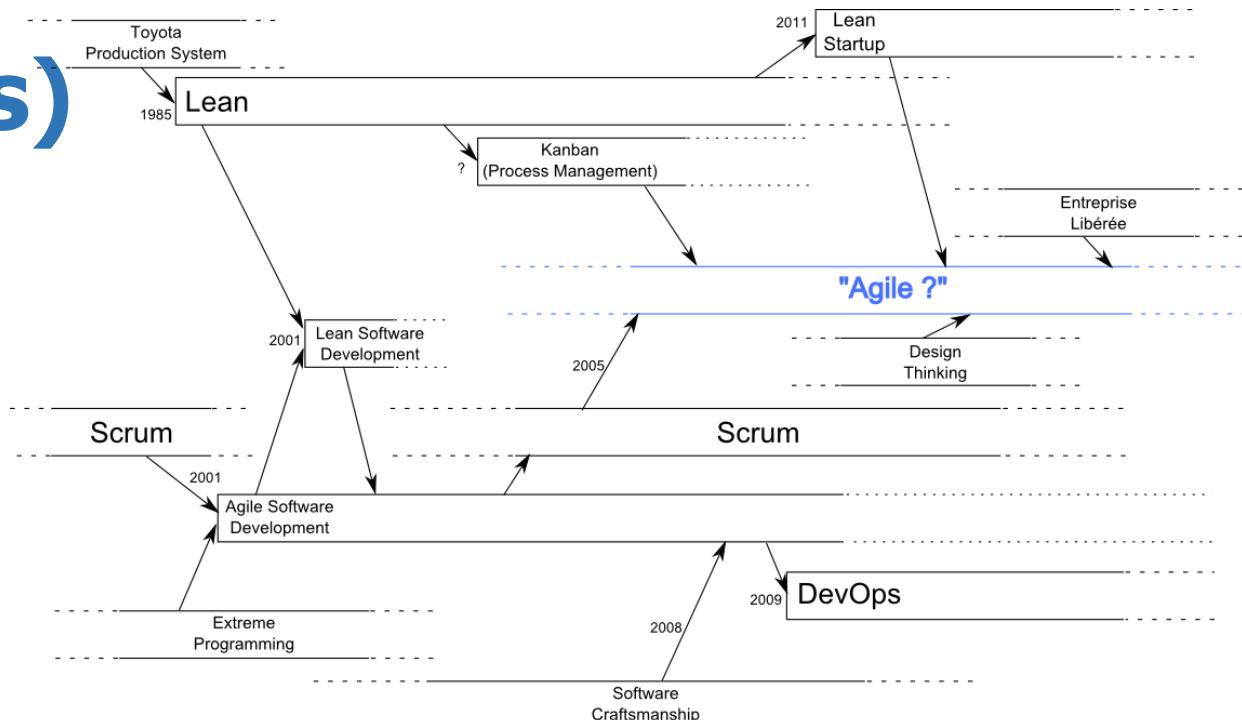
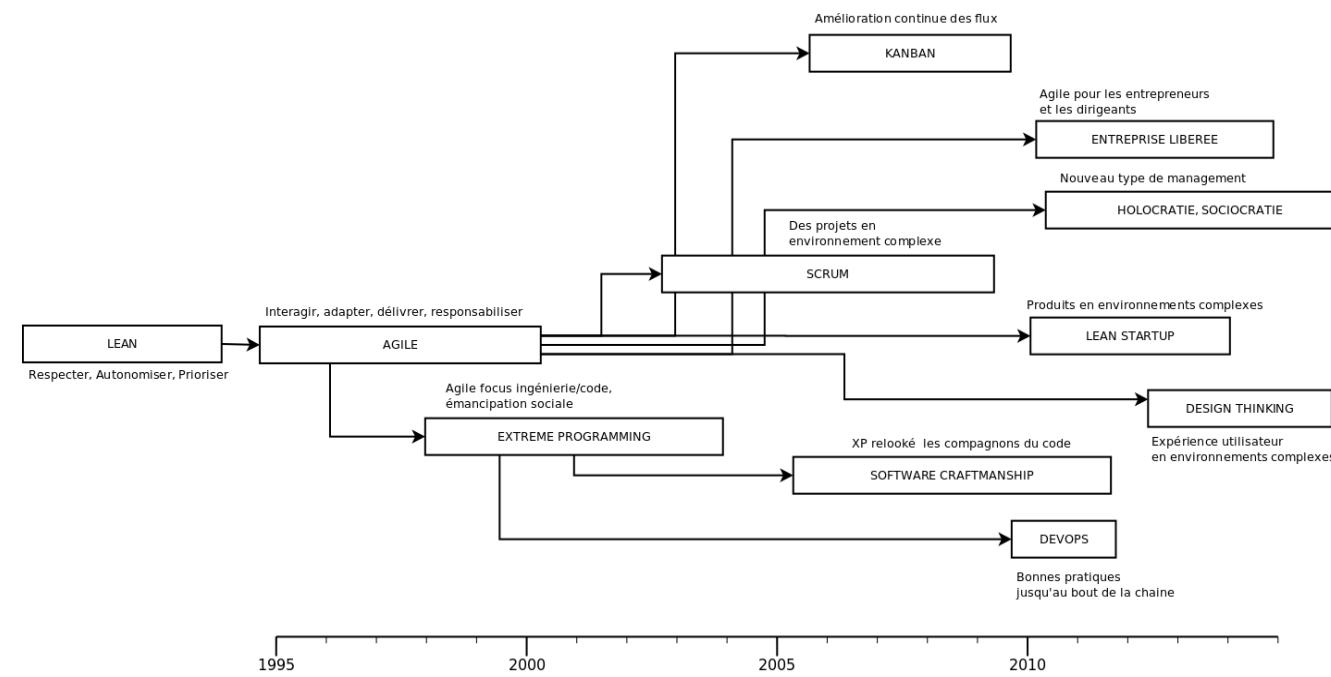
Les mouvements agiles du mouvement Agile

<http://www.areyouagile.com/2015/08/mouvement-agile>

un seul et unique "mouvement agile"

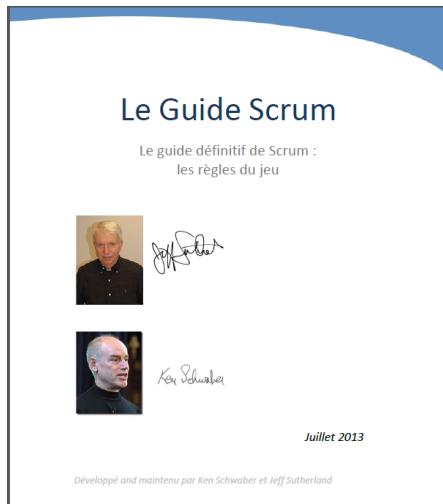
qui vient du lean et se décline en de multiples sensibilités ?

OU



« Ensemble de flux qui, potentiellement et ponctuellement, convergent vers quelque chose de commun, tout en gardant leur mouvement propre. S'il y a une "racine" commune à l'ensemble des approches agiles, elle sera dans le futur ». <http://agilitateur.azeau.com/post/2015/08/16/Mouvement%28s%29-Agile%28s%29>

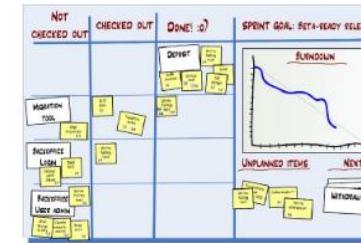
# Quelques lectures de référence sur Scrum



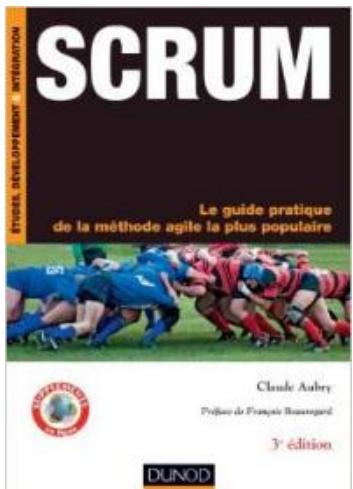
Le **scrum guide** de Ken Schwaber et Jeff Sutherland : <https://www.scrum.org/scrum-guide>  
(version française disponible)

## Scrum et XP depuis les Tranchées

Comment nous appliquons Scrum



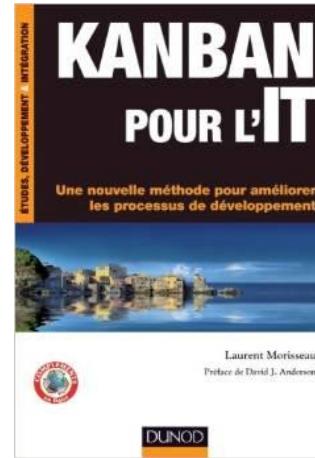
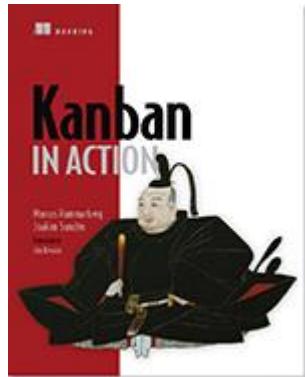
Écrit par :  
Henrik Kniberg



**Scrum : Le guide pratique de la méthode agile la plus populaire**  
<http://www.aubryconseil.com/pages/Livre-Scrum>

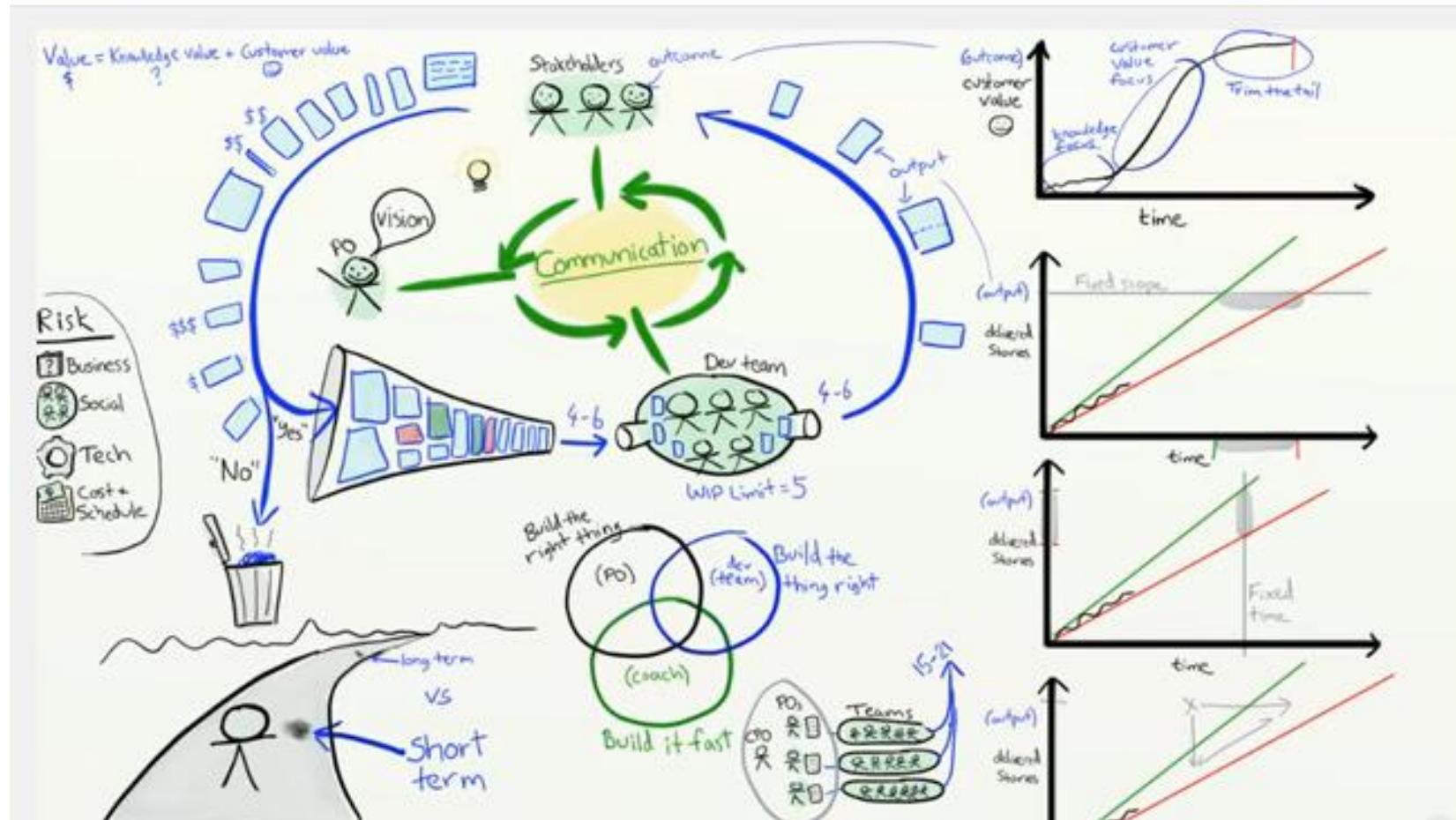
Blog Scrum, Agilité... et Rock'n roll : <http://www.aubryconseil.com/>

# Quelques lectures sur Kanban



- ✓ De nombreux liens sur :  
<http://www.crisp.se/kanban>

# La gestion du produit agile en deux mots



<http://www.youtube.com/watch?v=3qMpB-UH9kA>

# Activités pratiques du PO (1/2)

## L'Activité Principale du Product Owner

La création du PB et son maintien (ou « grooming » en anglais) sont les activités principales du PO, elle n'est pas facile car il lui faut récolter de nombreuses informations sur le produit :

- Identifier les attentes des utilisateurs et les bénéfices que le produit leur apportera
- Décrire avec le maximum de détails les utilisateurs et/ou clients du produit
- Identifier les fonctionnalités attendues et sélectionner celles qui apportent le plus de valeurs ou de bénéfices aux utilisateurs pour définir et planifier les releases/versions du produit
- Décrire chaque fonctionnalité retenue sous forme d'une User Story suffisamment petite pour être implémentée en 1 seule itération, sans oublier d'y associer les critères d'acceptation indispensables à sa bonne compréhension par l'équipe.
- Comprendre les Technical Story proposées par l'équipe de réalisation (besoins non fonctionnels mais indispensable – ex : Migration de version d'une base de données) et les Bug Story (ou « bugs connus »)
- Prioriser toutes les Story au sein du Product Backlog
- Maintenir le Product Backlog et chercher en permanence à maximiser la valeur métier pour les utilisateurs
- Accepter ou refuser les Story implémentées par l'équipe de réalisation

Extrait de : <http://www.agilex.fr/2013/02/product-owner-qui-es-tu/>

# Activités du PO dans Scrum (2/2)

## Autres Activités du Product Owner

Si l'élaboration et l'évolution du Product Backlog représentent des activités primordiales et consommatrices de temps, ce ne sont pas les 2 seules activités dévolues au PO, car il ne faut pas oublier les suivantes :

- Elaborer la Vision du Produit pour la version en cours
- Partager cette Vision avec l'équipe de réalisation et leur communiquer pourquoi le produit est utile
- Montrer aux décideurs / parties prenantes que cette vision est dans la ligne de la stratégie de l'entreprise
- Communiquer sur l'avancement de la réalisation du produit auprès du management et des utilisateurs
- Récolter les feedbacks des utilisateurs
- Répondre aux demandes de clarification émises par l'équipe de réalisation sur les Story en cours de réalisation durant l'itération.
- Contribuer aux réunions Scrum avec l'équipe de réalisation et le Scrum Master
- Réaliser ou organiser le déroulement des Tests Utilisateurs/Métier de la version
- Evaluer le fonctionnement des versions précédentes mises en production et en cours d'utilisation
- Mener une réflexion stratégique préparatoire des versions à venir

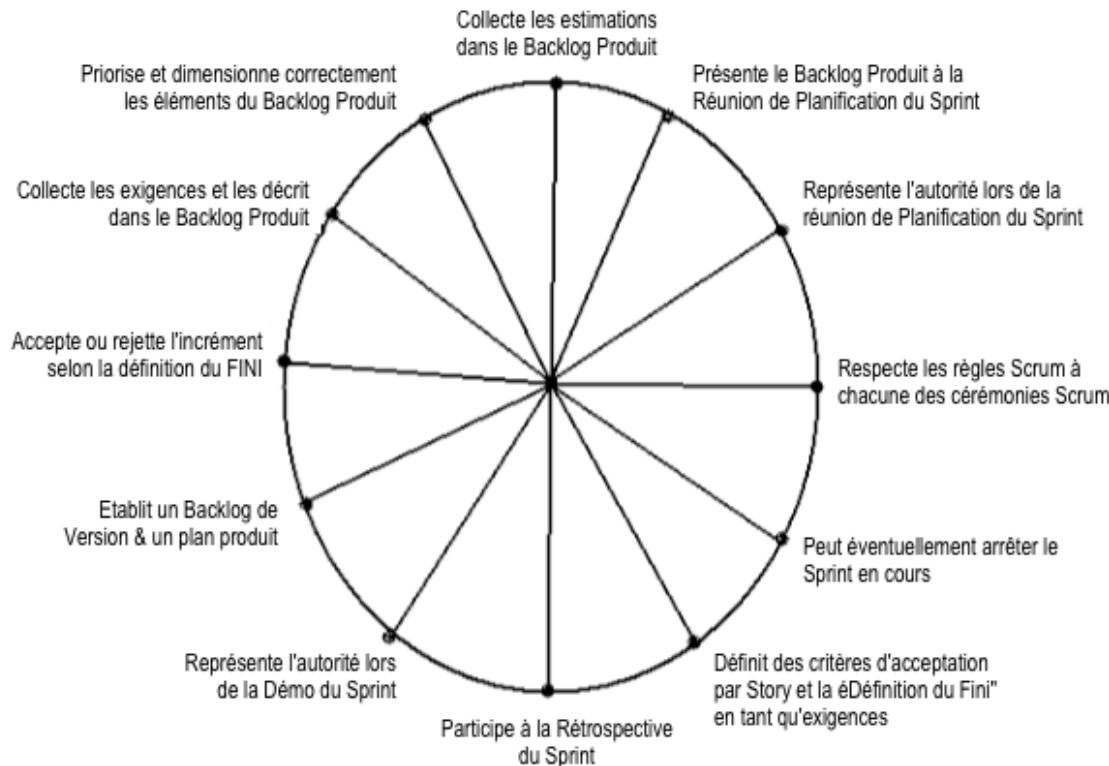
Et même parfois

- Assurer la formation des utilisateurs
- Promouvoir le produit auprès d'autres instances (Directions métier, Filiales ...)
- Evaluer les opportunités du marché
- Vendre le produit à des clients et ramener des commandes fermes !

# Carte des pouvoirs du Product Owner

*New Technology Solutions, Inc.*

## CARTE DES POUVOIRS DU PRODUCT OWNER



(c) 2010 New Technology Solutions Inc [www.NewTechUSA.com](http://www.NewTechUSA.com)



<http://www.NewTechUSA.net>

*Enabling Enterprise Agility*

# Carte des pouvoirs du Scrum Master

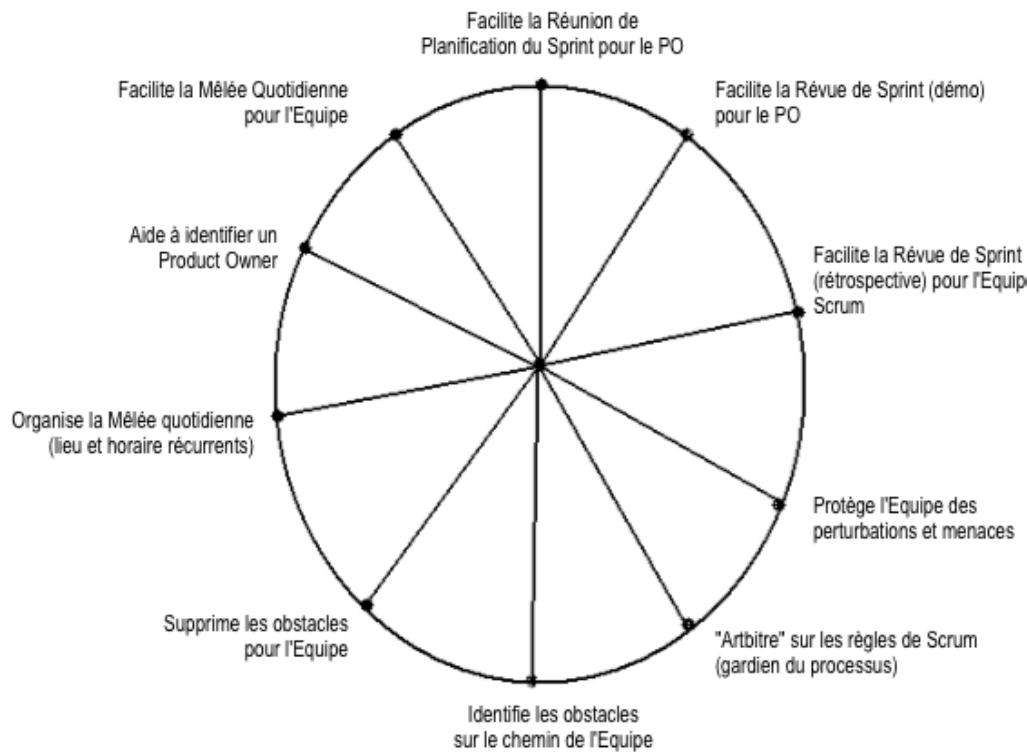


**New Technology Solutions, Inc.**

<http://www.NewTechUSA.net>

*Enabling Enterprise Agility*

## CARTE DES POUVOIRS DU SCRUM MASTER



(c) 2010 New Technology Solutions Inc [www.NewTechUSA.com](http://www.NewTechUSA.com)

# Carte des pouvoirs de l'équipe de développement

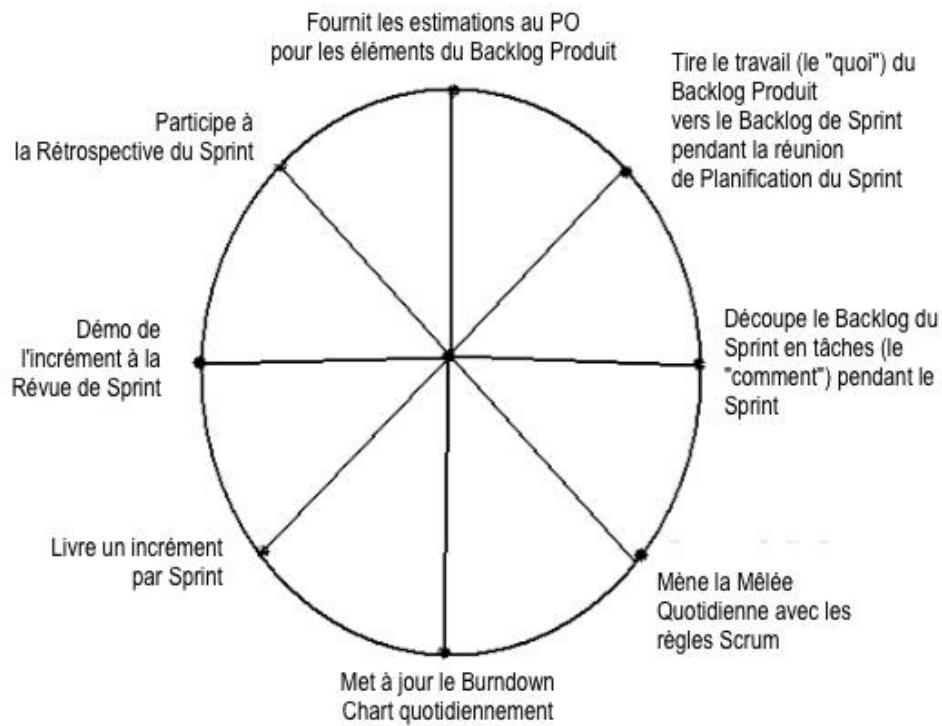


**New Technology Solutions, Inc.**

<http://www.NewTechUSA.net>

*Enabling Enterprise Agility*

## CARTE DES POUVOIRS DE L'ÉQUIPE



(c) 2010 New Technology Solutions Inc [www.NewTechUSA.com](http://www.NewTechUSA.com)

# Le développement agile en 2015 (outils)

## Top 5 Agile Techniques



9<sup>th</sup> State of Agile Survey 2015 (USA et Europe) par VersionOne

Extraits : <https://www.versionone.com/pdf/state-of-agile-development-survey-ninth.pdf>

## GENERAL TOOL USES & PREFERENCES

	CURRENT TOOL USAGE		FUTURE PLANS TO USE	
	2014	2013	2014	2013
Bug tracker	80%	83%	10%	5%
Taskboard	79%	81%	11%	6%
Spreadsheet	72%	68%	5%	3%
Wiki	68%	71%	12%	6%
Agile project management tool	65%	66%	20%	10%
Unit test tool	65%	65%	21%	12%
Automated build tool	65%	69%	20%	12%
Continuous integration tool	55%	57%	26%	14%
Kanban board	52%	43%	15%	9%
Traditional project management tool	51%	49%	7%	4%
Requirements management tool	50%	47%	22%	10%
Release/deployment automation tool	48%	47%	32%	14%
Index cards	41%	44%	10%	6%
Project & portfolio management (PPM) tool	37%	22%	24%	11%
Automated acceptance tool	35%	33%	39%	19%
Story mapping tool	34%	47%	29%	14%
Refactoring tool	29%	33%	26%	11%
Customer idea management tool	22%	21%	28%	11%