PRATIQUES DE L'AGILITÉ

smartview
conseil et formation

PABLO PERNOT - SMARTVIEW
http://www.areyouagile.com
https://speakerdeck.com/u/pablopernot
http://www.smartview.fr
http://convergenc.es

@pablopernot

© Scott Adams, Inc./Dist. by UFS, Inc.

version 2.1.15 - décembre 2012

Programme

JOUR 1

Les raisons et origines de l'agilité, valeurs & principes

Filiation et comparaison des principales méthodes agiles : Lean, XP, Scrum, KanBan

La pensée LEAN
- Respect des personnes
- Amélioration continue

SCRUM

- Aperçu de Scrum,
- Les acteurs de Scrum
- Développement itératif
- Timebox
- Communication, interaction

Atelier : Coin Toss, Offing the offsite

Pratiques d'expression du besoin
- Délivrer de la valeur
- Les User Stories
- Personas
- Backlog
- Notion de "fini"

Ateliers : Prune the tree, Open-ended specifications, Backlog

JOUR 2

Pratiques d'estimation et de planification
- Estimation, Planification
- Cycle de vie / Itérations
- Conception émergente

Ateliers : Planning poker, Wall planning poker, Marshmallow challenge

Pratiques quotidiennes et Pilotage
- Visualisation et "radiateurs" d'information
- Les burndown/up charts
- Les standups

Atelier : Scrum from hell

Pratiques de fin d'itération et de cycle
- Les revues
- Les rétrospectives

Atelier : Speedboat

EXTREME PROGRAMMING

Les pratiques d'ingénierie
- Dette technique
- Feeback
- Tests automatisés

PROGRAMME

JOUR 3

EXTREME PROGRAMMING

Pratiques d'ingénierie (suite)
- Refactoring
- Pair programming
- Intégration continue

Atelier : XP Game

KANBAN

Une marque de maturité ?

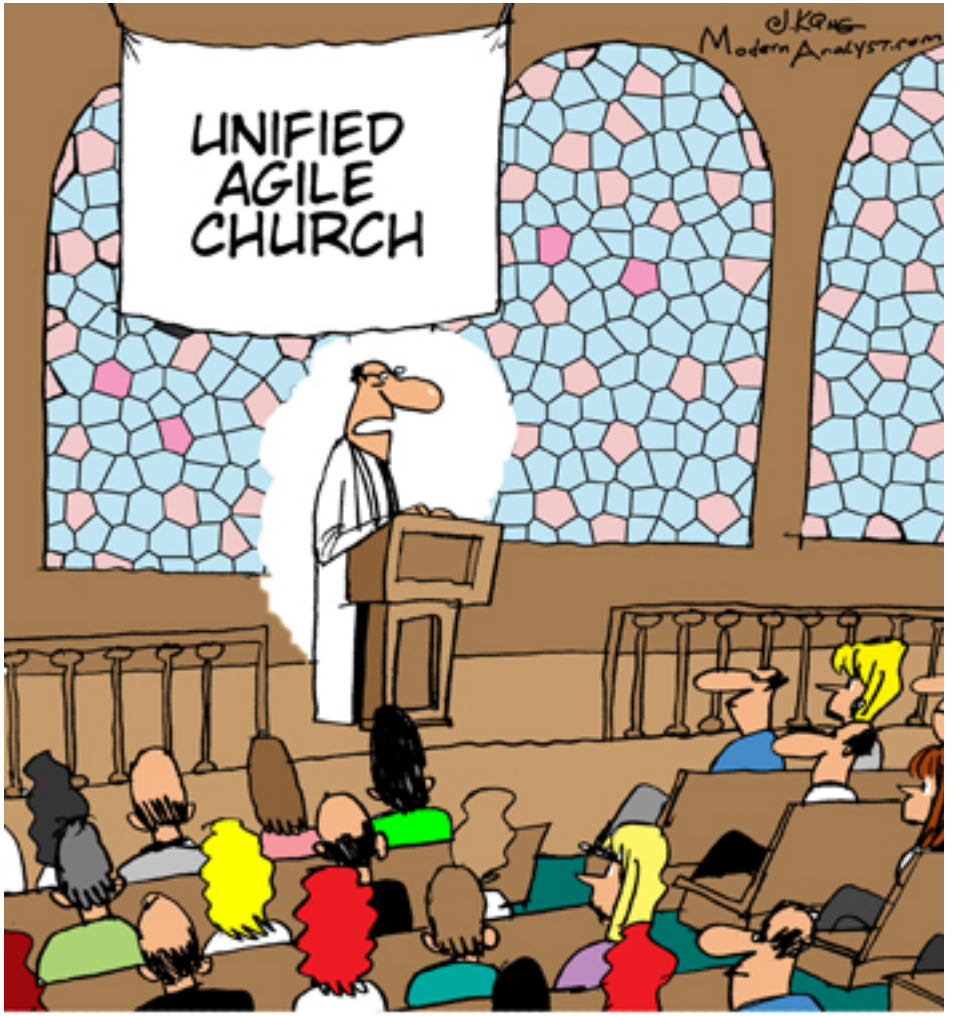
Evolution de certaines pratiques
- Mise en oeuvre de Kanban
- Visualiser le flux
- Classes de service
- Débats autour de Kanban

Atelier : KanBan Game

Pratiques de l'entreprise agile
- Passage à l'agilité, Conduite du changement
- Scalabilité, Management, Contractualisation

Atelier : leadership

Conclusion



"Repent of your Traditional sins! Only through Agile you can save your projects!"

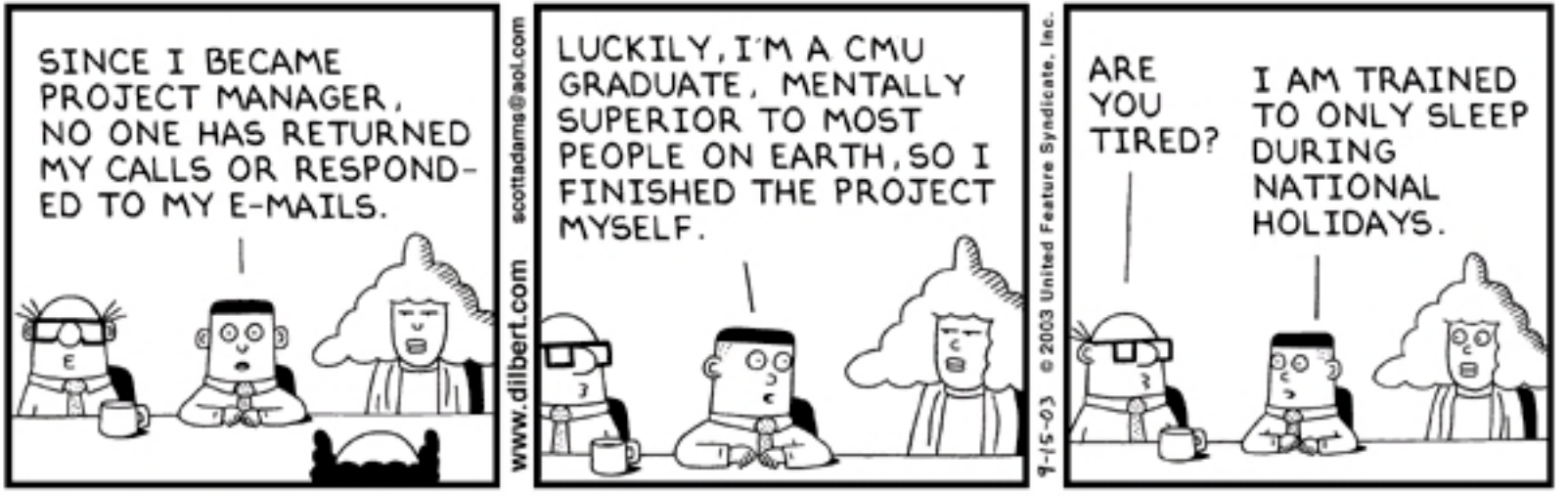
Présentations

Connaissez vous les méthodes agiles ? Avez-vous lu des livres sur les méthodes agiles ?

Votre organisation implémente-t-elle déjà une méthode agile ? Comment travaillez vous actuellement sur vos projets ? Pourquoi marchent-ils ? Pourquoi échouent-ils ?

Quels sont les problèmes auxquels vous êtes régulièrement confrontés ? Avez vous cherché des pistes d'améliorations ?

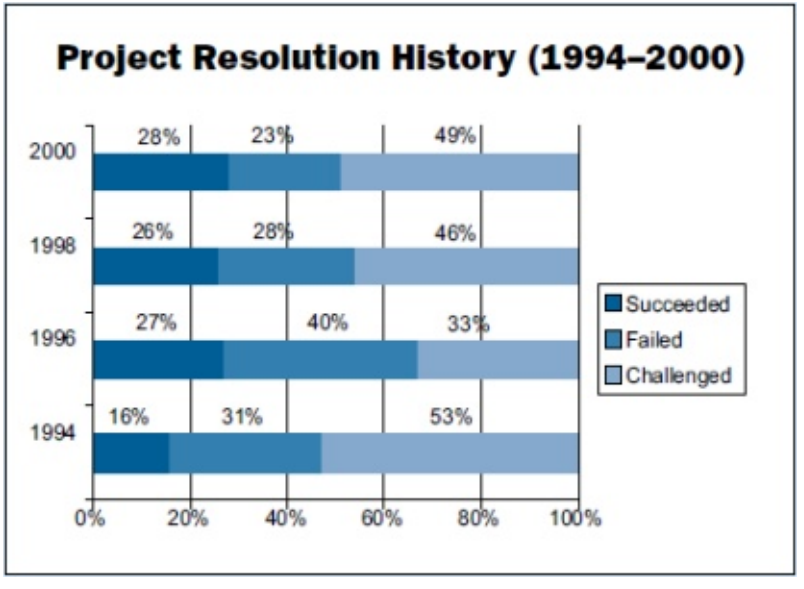
Pourquoi souhaitez-vous implémenter l'agilité au sein de votre organisation ?

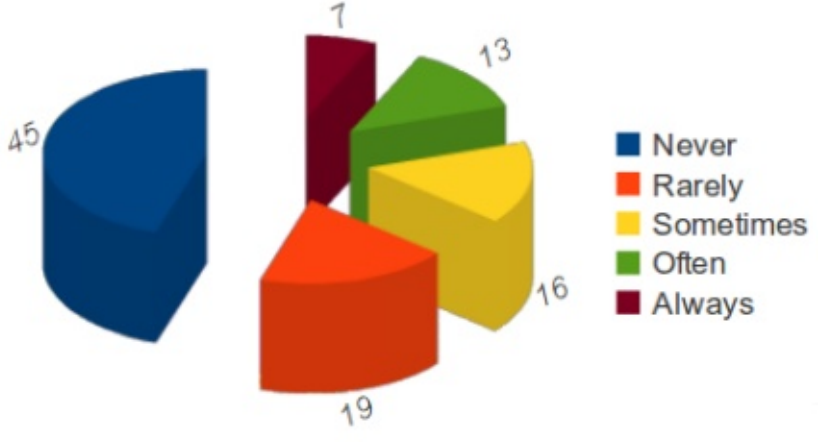


L'AGILITÉ AUJOURD'HUI

Les raisons de l'agilité

Le fameux rapport Chaos du Standish Group...





Principales causes des échecs

Mauvaise compréhension du besoin : 51%
Estimation et planification déficiente : 48%
Technologies mal maîtrisées : 45%

Facteurs clefs du succès

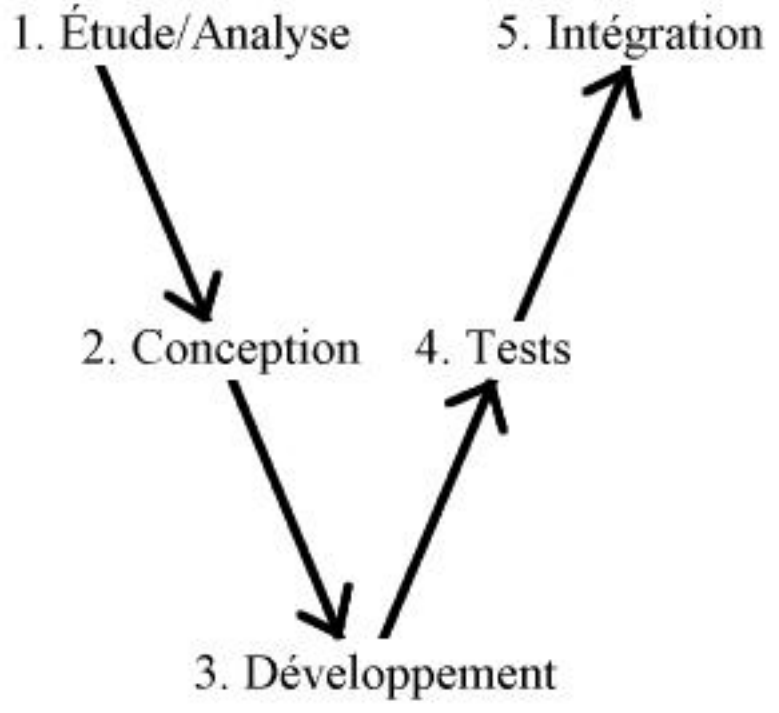
Implication des utilisateurs
Soutien par le management
Objectifs business clair
Périmètre optimisé

Un constat sévère

64% des fonctionnalités développées sont peu ou pas utilisées...

Source : http://www.cs.nmt.edu/~cs328/reading/Standish.pdf

Le cycle en V, traditionnel, et waterfall



Mais qu'est ce qui ne fonctionne pas avec ce modèle ?

La fin du cycle en V classique

L'immuabilité des spécifications et des besoins fonctionnels
- Ce qui sera délivré sera en partie obsolète, pensez aux marines ! **

L'effet tunnel !
- Feedback trop tardif : toute modification va coûter très cher

La sectorisation des intervenants
- Les équipes s'incriminent entre elles
- Insatisfaction des équipes

On oublie où se trouve la valeur du projet et on se concentre sur le respect à la lettre des spécifications (or elles sont souvent mal comprises, et surtout elles changent !) et de la documentation

On ne délivre pas assez de valeur au client

Nécessité de documenter de façon détaillée tous les éléments du projet

** Source : Mary Poppendieck, Lean Software Development, an agile toolkit

4 VALEURS AGILES

Pour répondre à ce problème est créé en 2001 le manifeste Agile. Rédige par 17 experts qui se dressent contre l'échec des cycles en cascade, Il propose 4 valeurs fondamentales, et 12 principes

La réactivité face au changement plutôt que le suivi d'un plan

l'interaction avec les personnes plutôt que les processus et les outils

un produit opérationnel plutôt qu'une documentation pléthorique

La collaboration avec le client plutôt que la négociation de contrat

On parle de rituels, la photo du manifeste parait mystique, mais ce n'est pas une secte ! ;)
http://fr.wikipedia.org/wiki/Manifeste_agile

Les 12 principes de l'agilité

Notre première priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels utiles.

Le changement est accepté, même tardivement dans le développement. Les processus agiles exploitent le changement comme avantage compétitif pour le client.

Livrer fréquemment une application fonctionnelle, toutes les deux semaines à deux mois, avec une tendance pour la période la plus courte.

Les experts métier et les développeurs doivent collaborer quotidiennement au projet.

Bâtissez le projet autour de personnes motivées. Donnez leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail.

La méthode la plus efficace pour transmettre l'information est une conversation en face à face.

Les 12 principes de l'agilité

Un logiciel fonctionnel est la meilleure unité de mesure de la progression du projet.

Les processus agiles promeuvent un rythme de développement soutenable.

Commanditaires, développeurs et utilisateurs devraient pouvoir maintenir le rythme indéfiniment.

Une attention continue à l'excellence technique et à la qualité de la conception améliore l'agilité.

La simplicité - l'art de maximiser la quantité de travail à ne pas faire - est essentielle.

Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto-organisent.

À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accorde et ajuste son comportement dans ce sens.

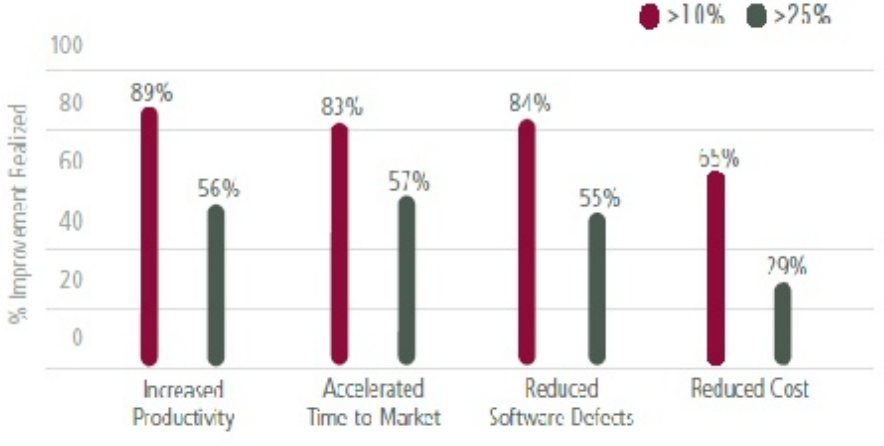
Les méthodes agiles sont matures

Plusieurs méthodes : XP, SCRUM, LEAN, etc.

Plus de 10 ans d'expérience

Une méthodologie reconnue :
par Google, Yahoo, Nokia, Microsoft, IBM, Oracle, MySpace, Adobe...

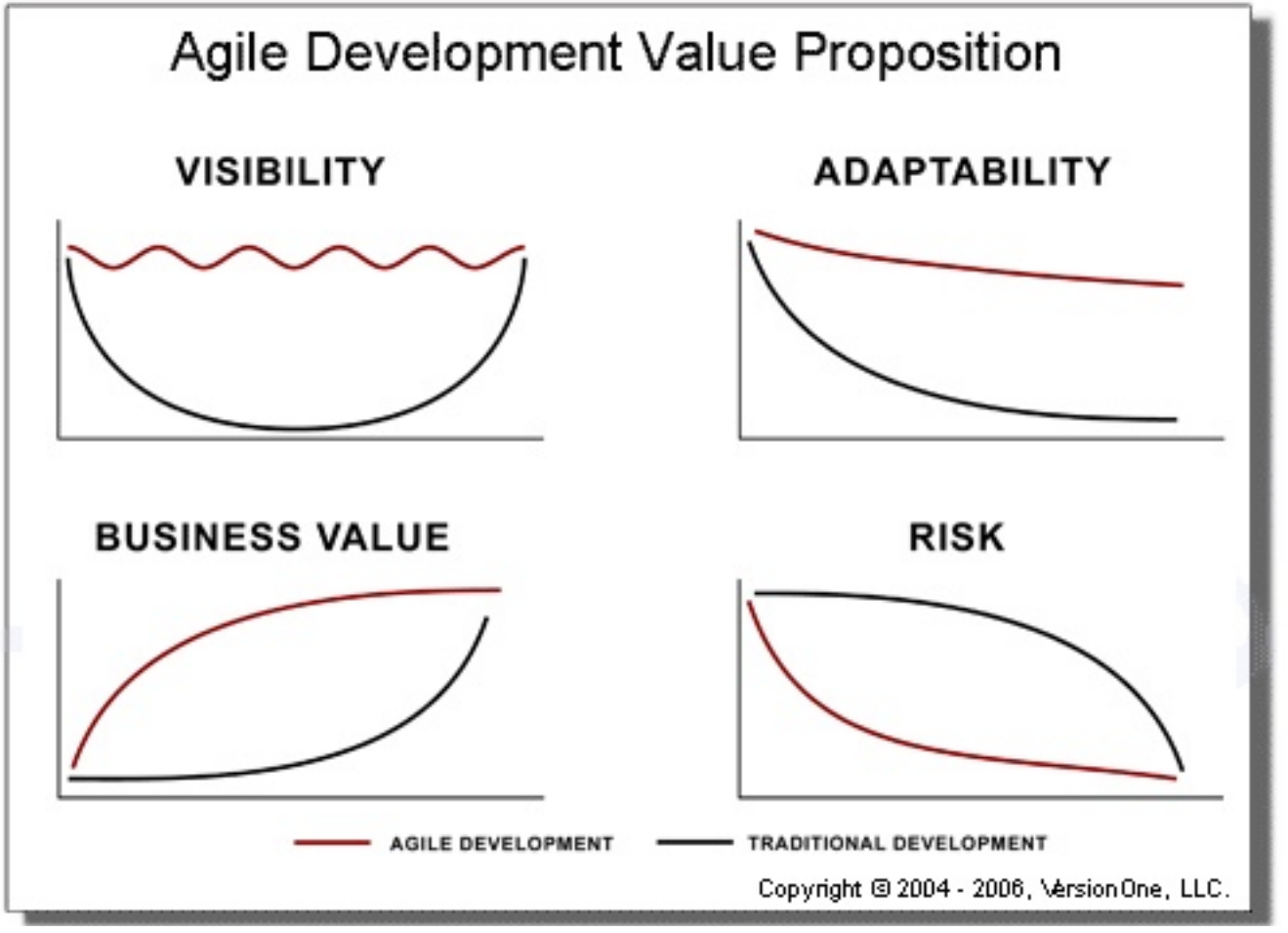
En 2008, l'étude du drdobbs.com indique que 69% des entreprises font de l'agile, et parmi celles qui ne font pas d'agile, 15% d'entre elles estiment démarrer l'année suivante



	% Improvement Realized			
	>25%	25%	10%	NONE
Increased Productivity	25%	31%	33%	11%
Accelerated Time-to-Market	30%	26%	26%	17%
Reduced Software Defects	30%	25%	29%	16%
Reduced Cost	13%	17%	36%	35%

** Etude Agile Adoption Rate Survey Results: February 2008 http://www.ambysoft.com/surveys/agileFebruary2008.html*
*** 3rd Annual Survey 2008 "The state of Agile development" http://pm.versionone.com/whitepaper_AgileSurvey2008.html*

Bénéfices proposés par l'agilité



Pourquoi cela marche ?

Priorité à ce qui a le plus de valeur, à ce qui est le plus important

Démarche itérative, incrémentale et adaptative

Des interactions et de la communication

Un outillage compact et rapidement assimilable

De la visibilité

De la motivation et de la satisfaction dans les équipes

Un produit opérationnel très tôt

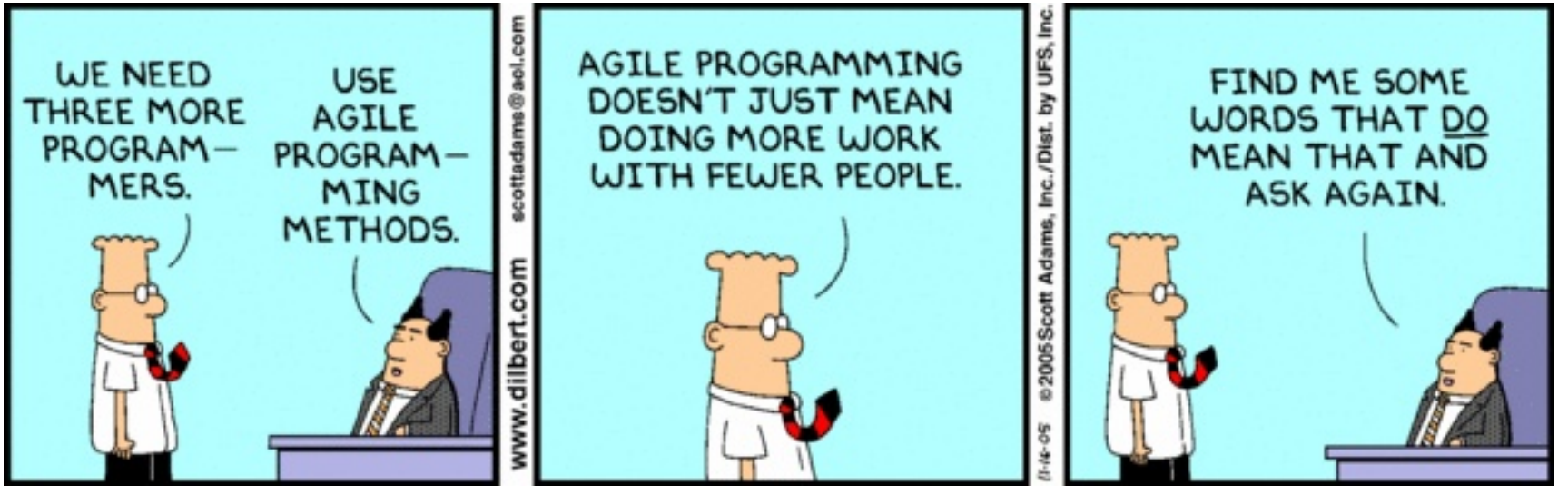
Une réactivité face au changement

Mais ce n'est pas

L'absence de règle :
- La discipline est indispensable
- Les processus sont indispensables

L'absence de documentation
- Il faut des spécifications si elles ont de la valeur

Magique
- Cela ne fonctionne pas dans tous les contextes
- Il n'y a pas de checklist Scrum !



LA PENSÉE LEAN

Le modèle de production Toyota

Amélioration continue
Respect des personnes
Remettre tout en cause
Embrasser le changement

Source : http://www.leanprimer.com/downloads/lean_primer.pdf

Taiichi Ohno
1912-1990

La base de la méthode Toyota est de ne pas se satisfaire du statu quo, vous devez constamment vous poser la question :
« Pourquoi faisons-nous ça ? ».

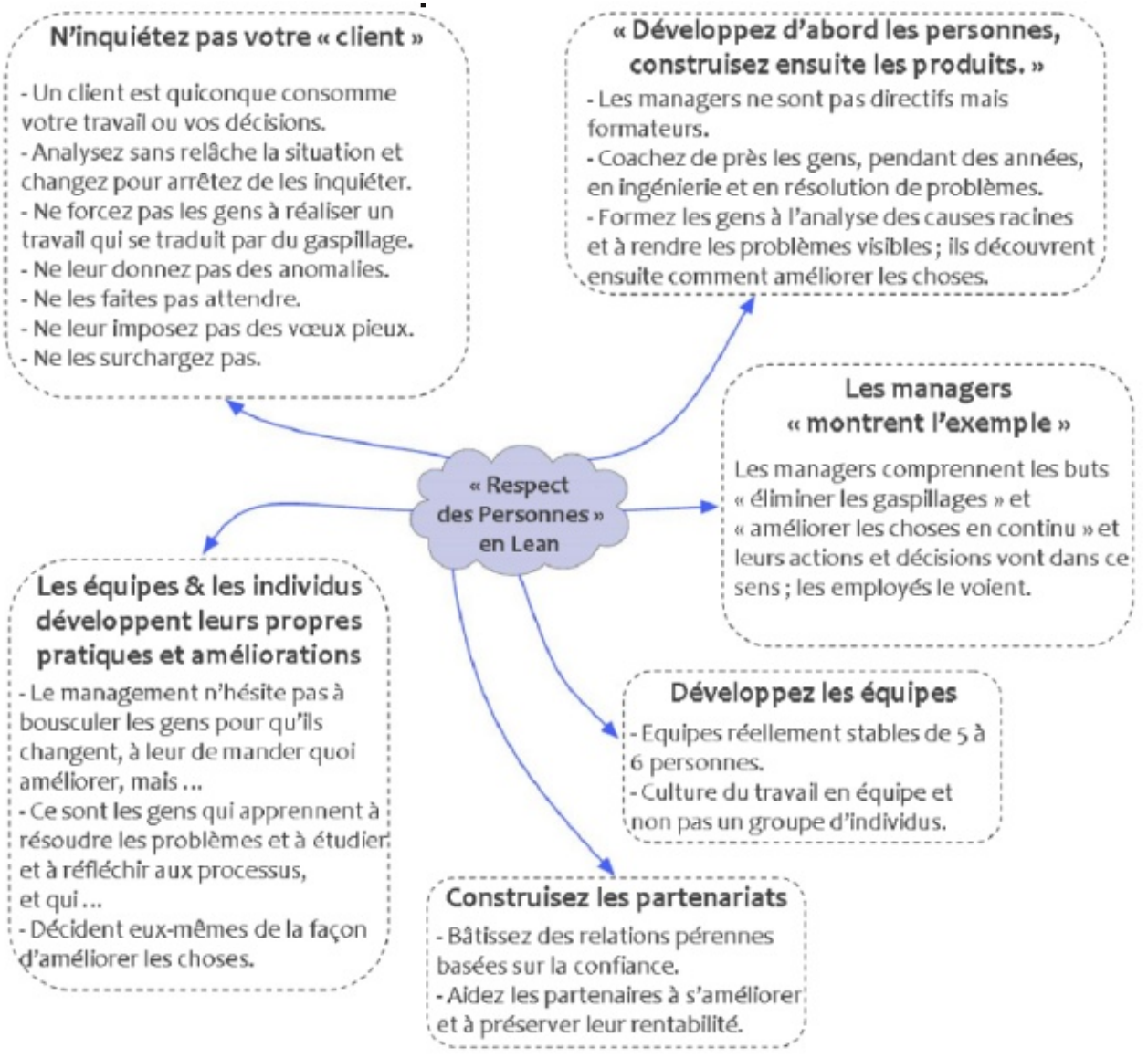
Source : http://www.fabrice-aimetti.fr/dotclear/index.php?post/2011/08/24/Lean-Primer

Une philosophie

Respect des employés
Une utilisation de toutes
les compétences des employés
Donner des responsabilités et avoir
confiance dans les employés

Source : Mary Poppendieck, Lean Software Development

Les piliers du Lean : Respect des personnes



Source : http://www.leanprimer.com/downloads/lean_primer.pdf
http://www.fabrice-aimetti.fr/dotclear/index.php?post/2011/08/24/Lean-Primer

Les piliers du Lean : Amélioration continue

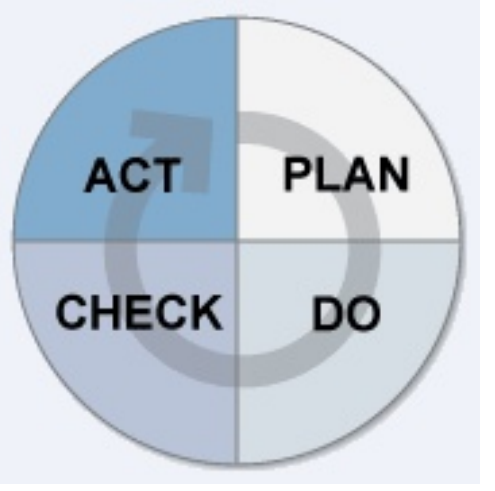
Allez & Observez

Kaizen

Shu-Ha-Ri
Valeur
Gaspillage
Petits ajustements incessants

Plein de mots japonais pour briller en société (ou faire consultant)

Kaizen : amélioration continue
Gemba : Là où se trouve la réalité
Muda : forme de gaspillage
Kanban : fiche, étiquette
Yokoten : Diffusion horizontale des connaissances
Jidoka : automatisation avec une touche humaine
Obeya : grande salle (war room des projets agiles...)



http://fr.wikipedia.org/wiki/Roue_de_Deming

Source : http://www.leanprimer.com/downloads/lean_primer.pdf
http://www.fabrice-aimetti.fr/dotclear/index.php?post/2011/08/24/Lean-Primer

Eliminer les sources de gaspillage

Seven Wastes of Manufacturing*	Seven Wastes of Software Development
Inventory	Partially Done Work
Extra Processing	Paperwork
Overproduction	Extra Features
Transportation	Building the Wrong Thing
Waiting	Waiting for Information
Motion	Task Switching
Defects	Defects

Source : Mary Poppendieck, Lean Software Development

Principes Lean

Optimiser le système dans son ensemble
- Approche à long terme.

Travailler en mode flux
- Cycle court
- Livrer rapidement
- Visibilité régulière

Décider le plus tard possible
- Reporter la décision
- "Last Responsible Moment"

Lisser le travail

Maîtriser les standards

Management visuel simple

Technologie éprouvée

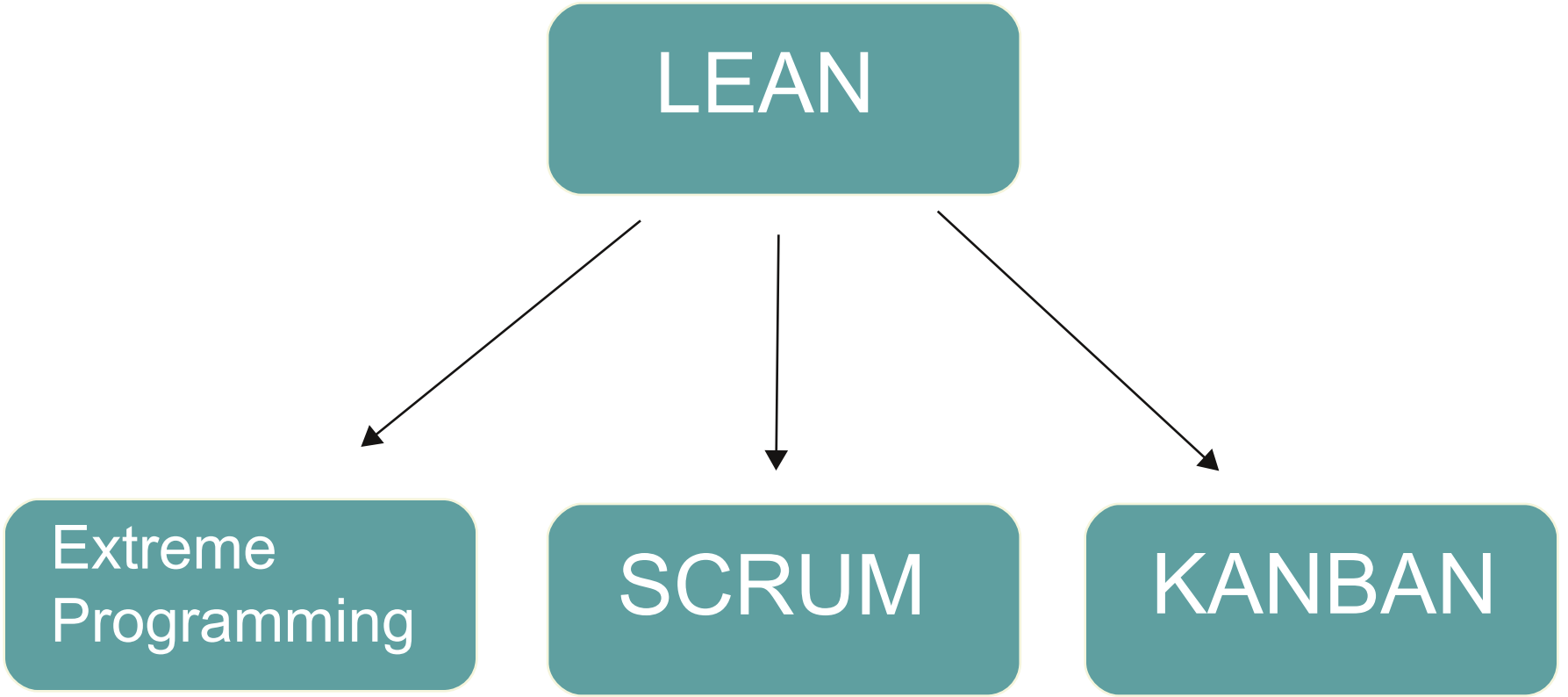
Responsabiliser l'équipe

Construire de la qualité intrinsèque
- Résoudre les problèmes

Eliminer les gaspillages

Montrer le plus tôt possible

LEAN, XP, SCRUM, KANBAN

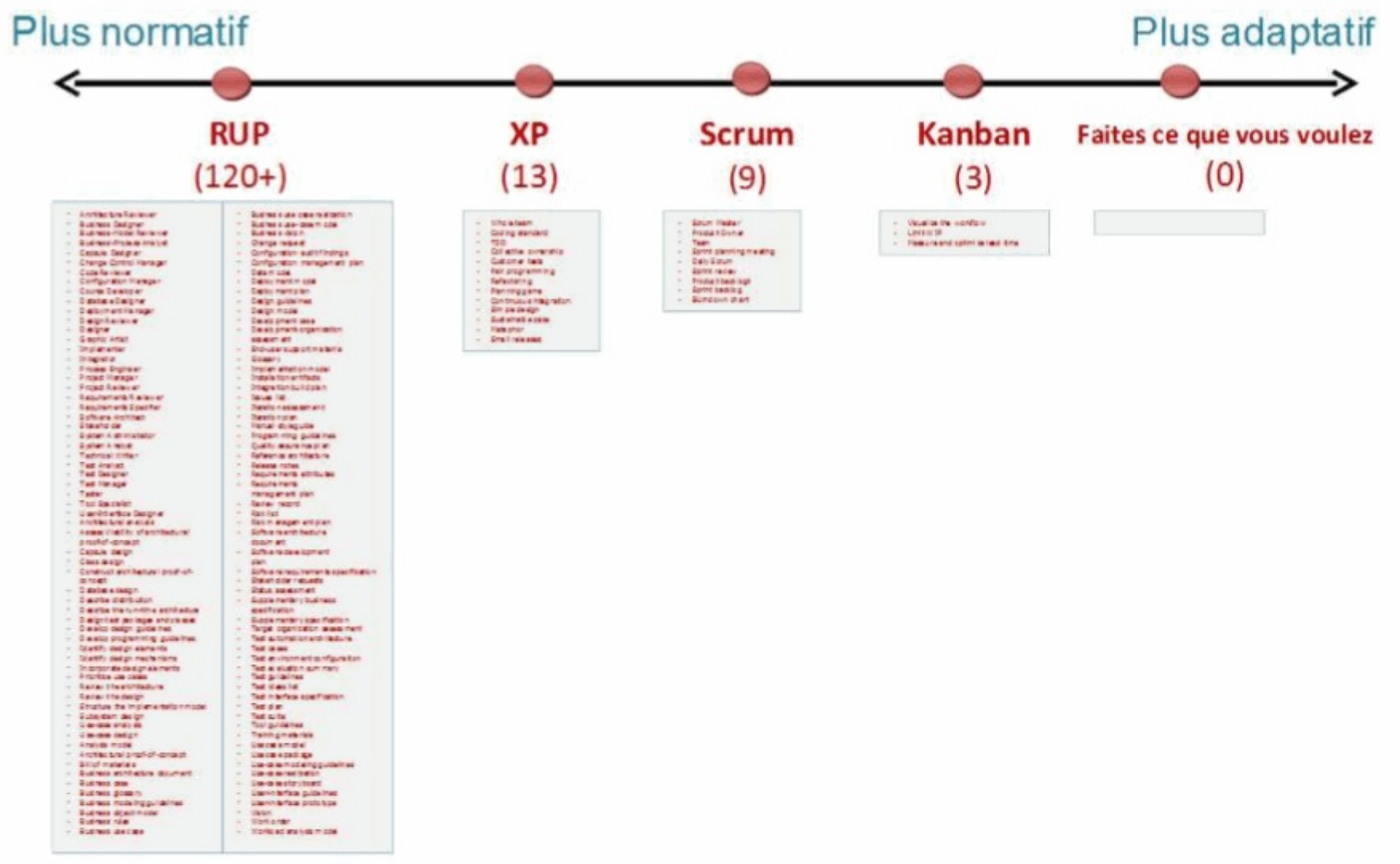


Pratiques d'ingénierie logicielles

Framework de développement logiciel

Système de gestion de flux de production

DES FAÇONS DE FAIRE PLUS OU MOINS NORMATIVES



Les valeurs de XP

Communication

Simplicité

Feedback

Courage

Respect

Les valeurs de Scrum

Transparence

Inspection

Adaptation

Aborder l'agilité et sa mise en oeuvre

SHU : Apprendre les fondamentaux

HA : Trouver les exceptions, trouver de nouvelles approches

RI : Tout redevient permis (on oublie la règle initiale, qui est digérée)

HUM HUM

Habituellement

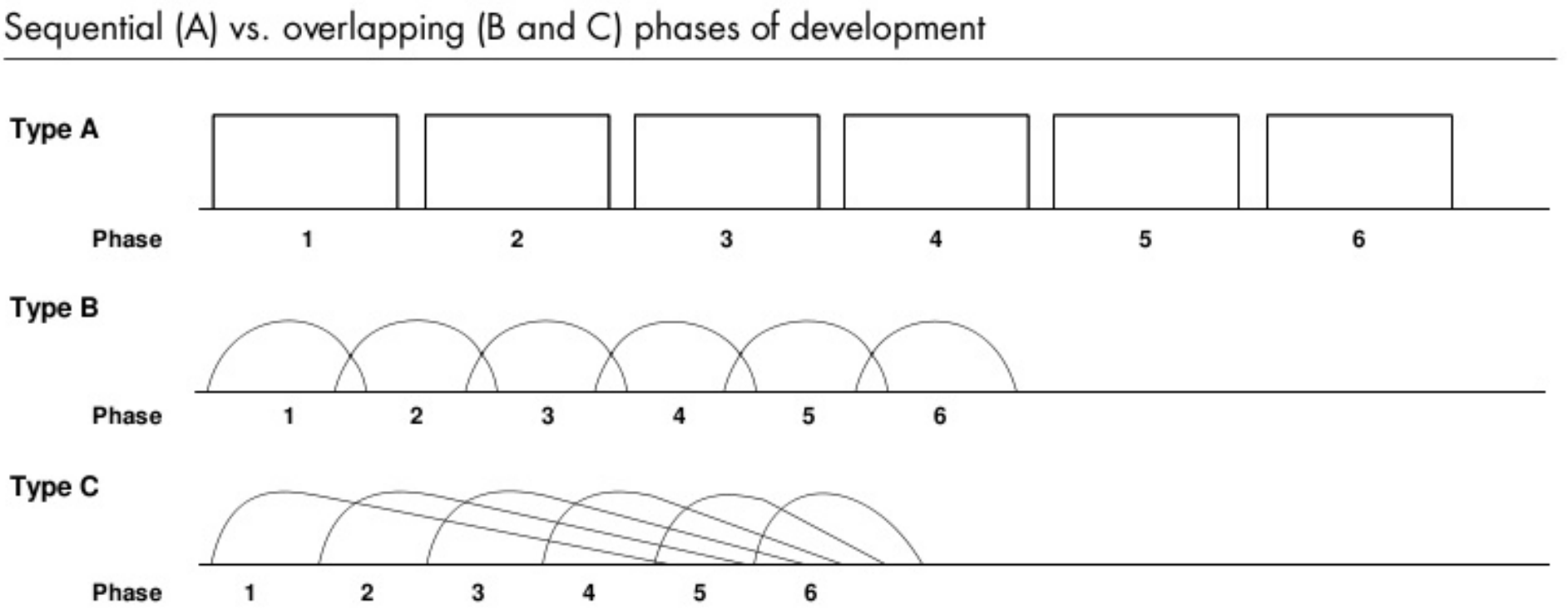
A ce moment dans la formation
on m'indique que c'est un peu
le monde des bisounours
l'agile...

passons à la pratique
a des choses plus concrêtes

Du développement itératif : Atelier "Coin Toss"

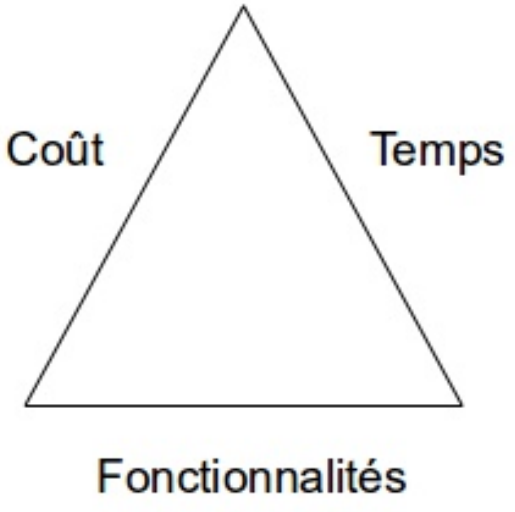
atelier
"COIN TOSS "

Du développement itératif



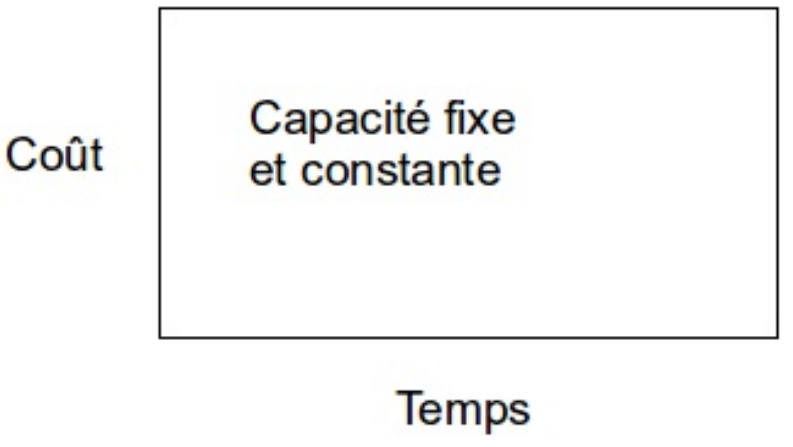
http://www.sao.corvallis.or.us/drupal/files/The%20New%20New%20Product%20Development%20Game.pdf
Source: “The New New Product Development Game”, Hirotaka Takeuchi and Ikujiro Nonaka,
Harvard Business Review, January 1986.

Qu'est ce qu'une "boite de temps" ? (timebox)



La répétition
d'une cadence bien maîtrisée

Yesterday weather ?
http://martinfowler.com/bliki/YesterdaysWeather.html



Les time-boxes dans Scrum

Sprint

Habituellement entre 2 et 4 semaines.

Sprint Planning Meeting

Pour 2 à 3 semaines de sprint : généralement 1 journée,
dont la moitié pour définir le contenu du sprint,
et l'autre moitié pour définir comment les fonctionnalités
sélectionnées vont être développées.

Sprint review

Généralement 4h< (la moitié de la durée du Sprint Planning Meeting). Plutôt 1 à 2h.

Sprint retrospective

Généralement 4h< (la moitié de la durée du Sprint Planning Meeting). Plutôt 1 à 2h.

Daily Scrum

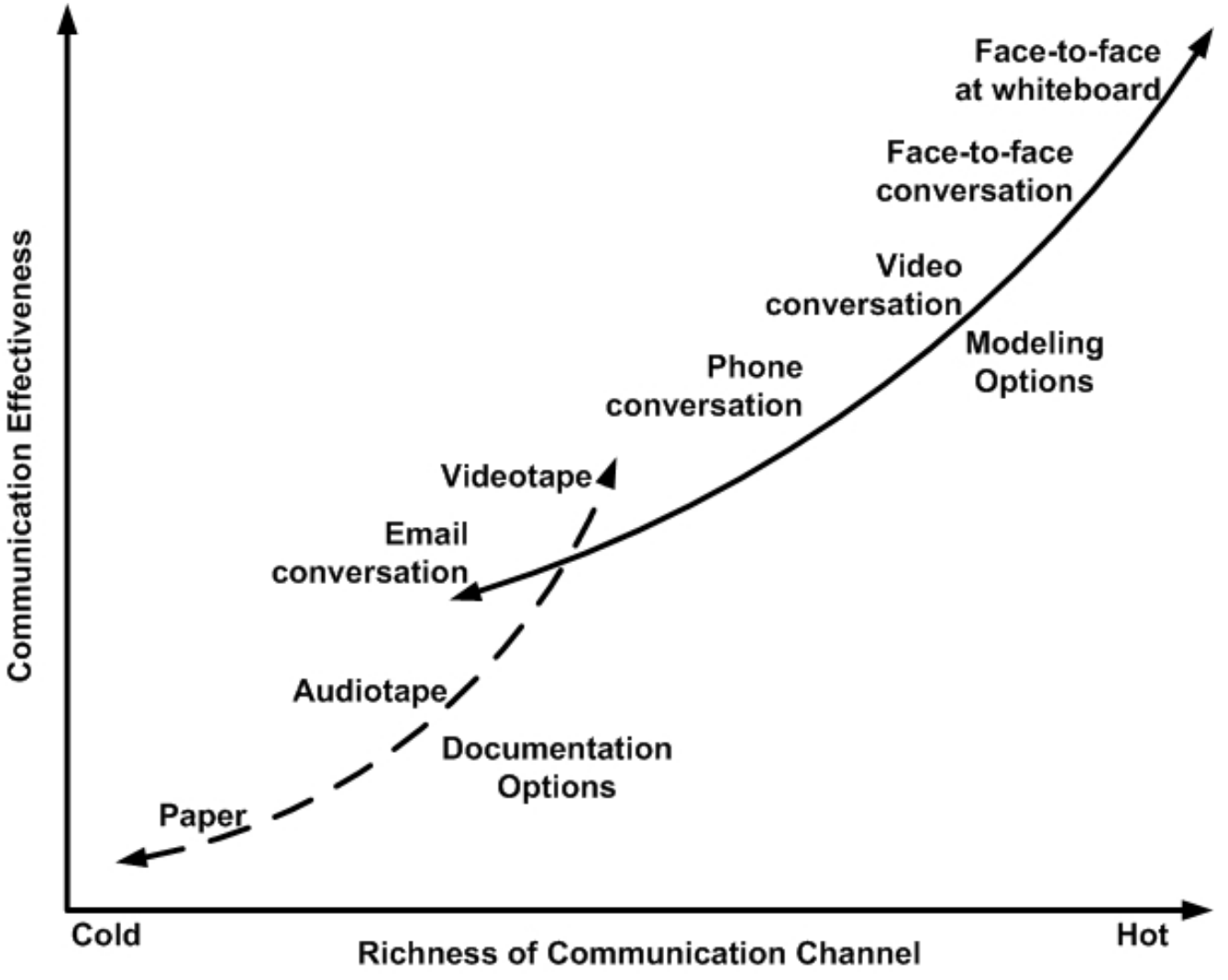
15Mn

WORKSHOP : "OFFING THE OFFSITE CUSTOMER"

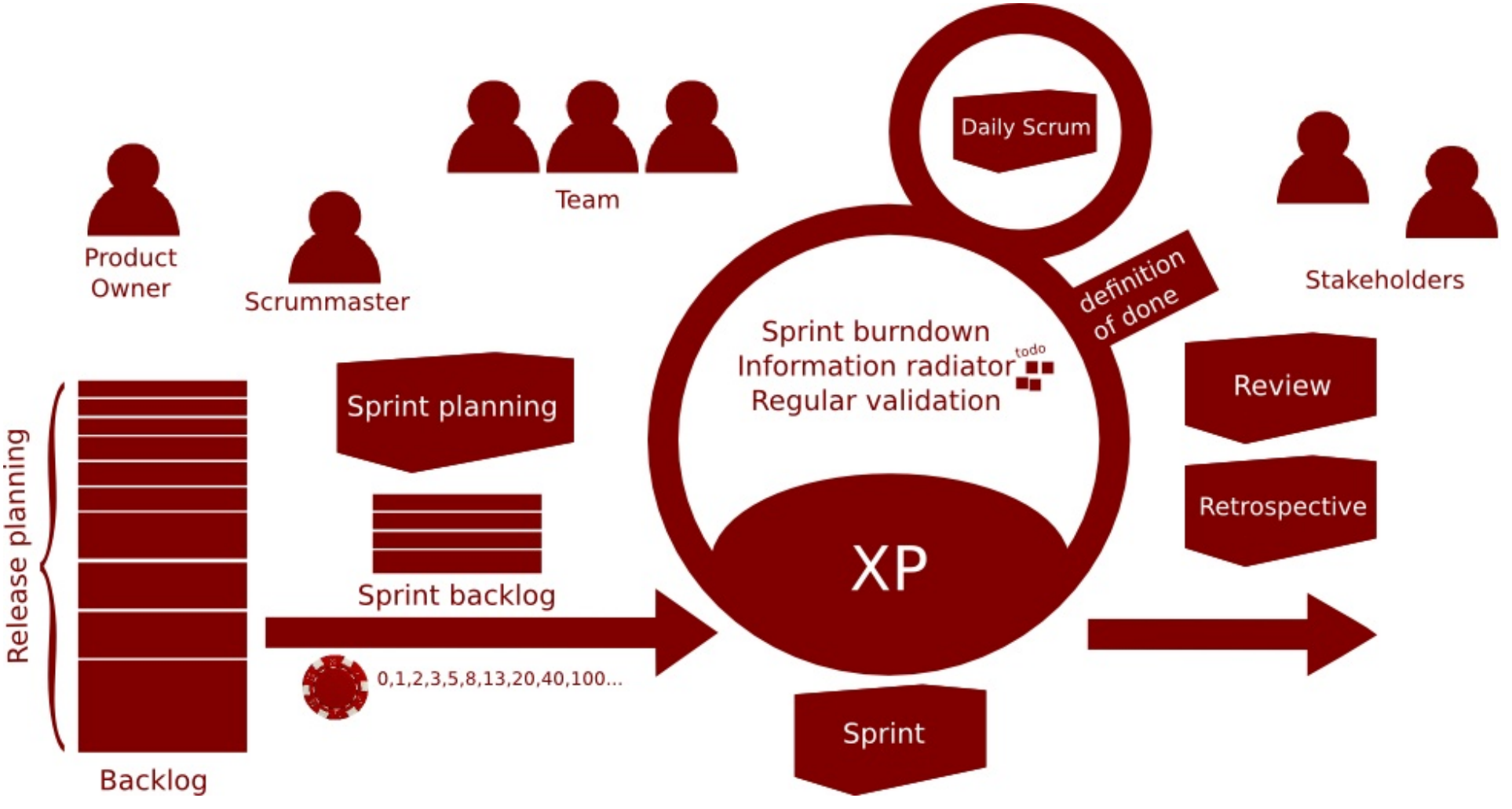
atelier
"Offing the offsite"

Source : James Shore
http://jamesshore.com/Presentations/OffingTheOffsiteCustomer.html

Modes de communication



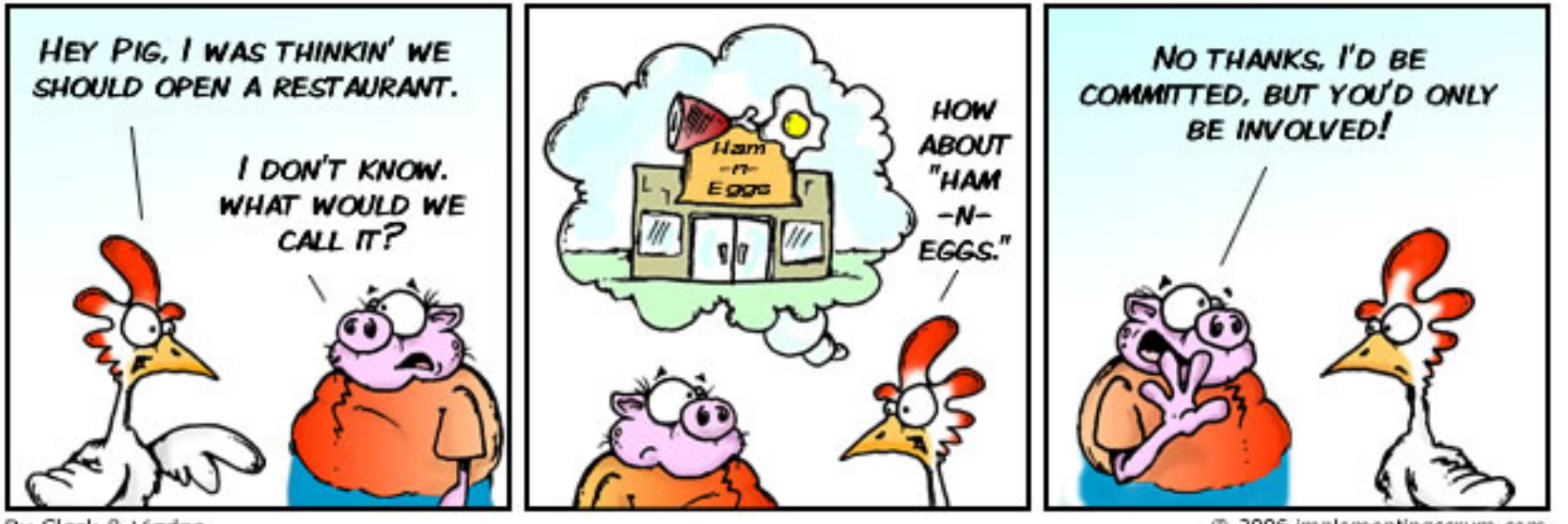
MISE EN PLACE D'UN PROJET SCRUM



Les acteurs SCRUM

Pig & Chicken

Entre les parties prenantes (stakeholders) et l'équipe Scrum, l'engagement est fondamental différent.



Le "Product Owner"

Décide des fonctionnalités du produit

Décide des dates de disponibilité des versions et de leurs contenus

Responsable de la valeur du produit

Priorise les fonctionnalités du produit en fonction de leurs valeurs ajoutées, du retour sur investissement, de leurs valeurs

Accepte ou rejette les résultats produits par l'équipe

Il doit être légitime et avoir assez de pouvoir pour pouvoir trancher quand cela est nécessaire (ce n'est pas un comité, c'est une seule personne)

Il doit être disponible et impliqué sur le projet (présence lors du sprint planning meeting, de la démo, de la rétrospective, des daily scrum)

L'ÉQUIPE

Généralement entre 5 et 9 membres (3 et 4 c'est bien, mais l'équipe doit être assez autonome (cross-functionnality)

Hétérogène (elle mêle architecte, développeur, testeur, analystes, graphistes, etc.)

Donne son accord aux objectifs de chaque itération

Spécifie les résultats à réaliser, elle s'engage et est responsable

Fonctionne de manière empirique et a le droit de faire tout ce qu'elle souhaite pour atteindre l'objectif de l'itération

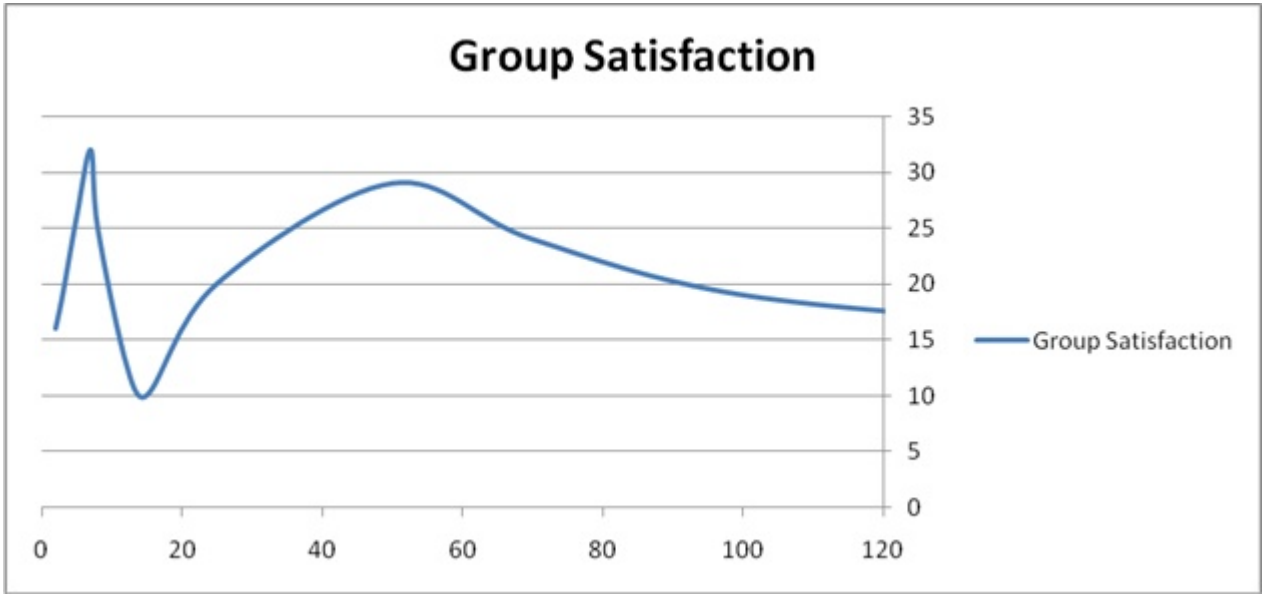
Autonome, elle fonctionne en autogestion

L'agile pense que nous sommes intelligents

Stable

Présente les résultats du travail de chaque itération au Product Owner
Il s'agit de résultats d'équipe

L'ÉQUIPE



inspiré de
http://www.leanessays.com/2011/02/before-there-was-management.html

L'équipe : modèle de Tuckman

Forming (formation de l'équipe)

Découverte et formation de l'équipe, définition des objectifs,

Storming

Confrontation et affrontement

Norming

L'équipe apprend à travailler ensemble

Performing

L'équipe fonctionne comme une unité très performante

http://en.wikipedia.org/wiki/Tuckman's_stages_of_group_development
http://www.areyouagile.com/2010/12/management-agile-le-modele-tannenbaum-schmidt/

L'équipe : modèle de Tannenbaum & Schmidt

1. Le manager décide et annonce la décision

2. Le manager décide et “vend” la décision au groupe

3. Le manager présente la décision avec des idées générales et invite à la réflexion

4. Le manager suggère une décision et invite à la réflexion

5. Le manager présente la situation, écoute les suggestions et décide

6. Le manager présente la situation, défini les contraintes et demande à l’équipe de décider

7. Le manager permet à l’équipe d’identifier le problème, développer les options, décider de la solution. Le manager est informé des contraintes par l’équipe.

http://www.businessballs.com/tannenbaum.htm
http://www.areyouagile.com/2010/12/management-agile-le-modele-tannenbaum-schmidt/

Le "Scrummaster"

S'assure que Scrum est bien appliqué

S'assure que l'équipe est opérationnelle et productive (et en bonne santé)

Facilite la coopération entre les différents acteurs (de l'équipe scrum ou des parties prenantes)

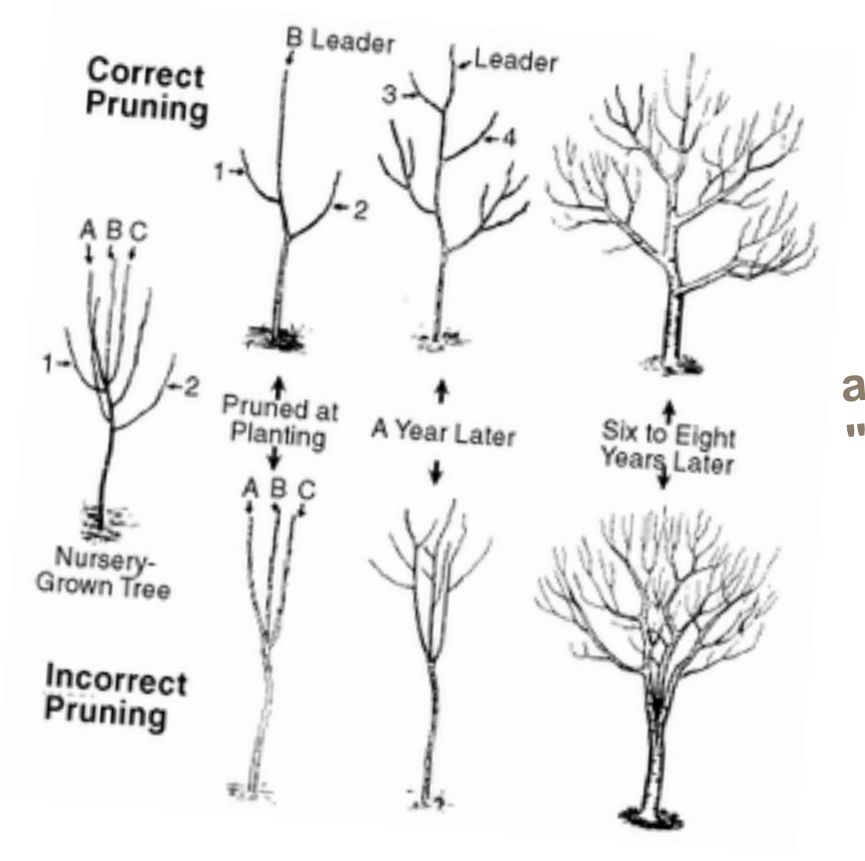
S'assure que rien ne gène l'avancée du travail, le cas échéant essaye de trouver la solution pour lever les blocages, obstacles visibles ou invisibles.

Réduit au maximum les perturbations extérieures avec l'équipe Scrum

Communique avec le management

S'assure que la qualité du produit n'est jamais remise en cause

Atelier : "Prune the tree"



atelier "Prune the tree"

Source : http://innovationgames.com/prune-the-product-tree/

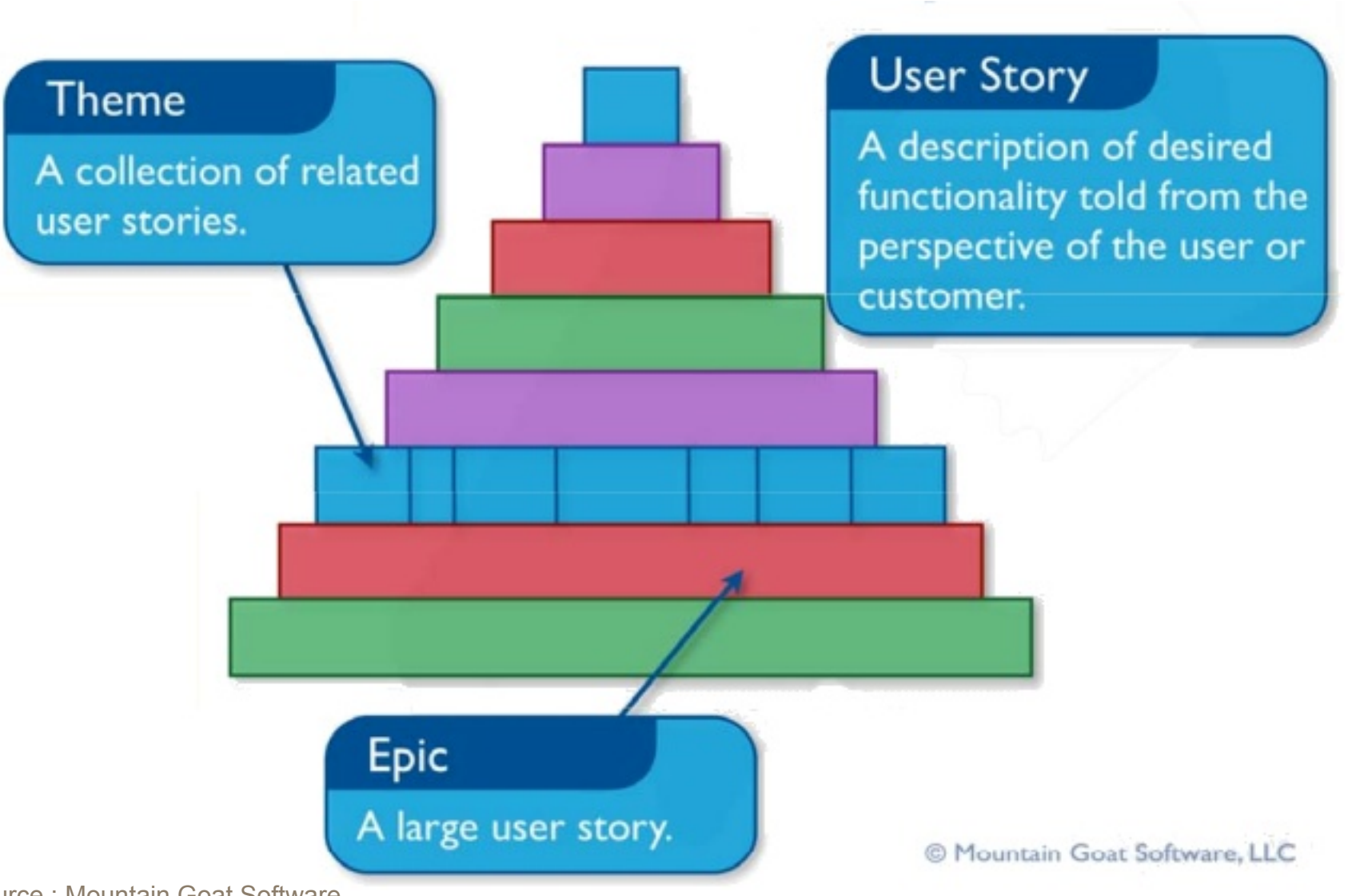
Délivrer de la valeur

La force de Scrum et de l'agile est de délivrer d'abord et avant toute chose ce qui a le plus de valeur (retour sur investissement). Cette valeur peut-être dans certains cas estimée (en monnaie).

Le Product Owner définit dans un premier temps une vision, une épopée concernant le produit. Puis il décline features/thèmes, cas d'utilisation ou histoires d'utilisateurs (User Story).

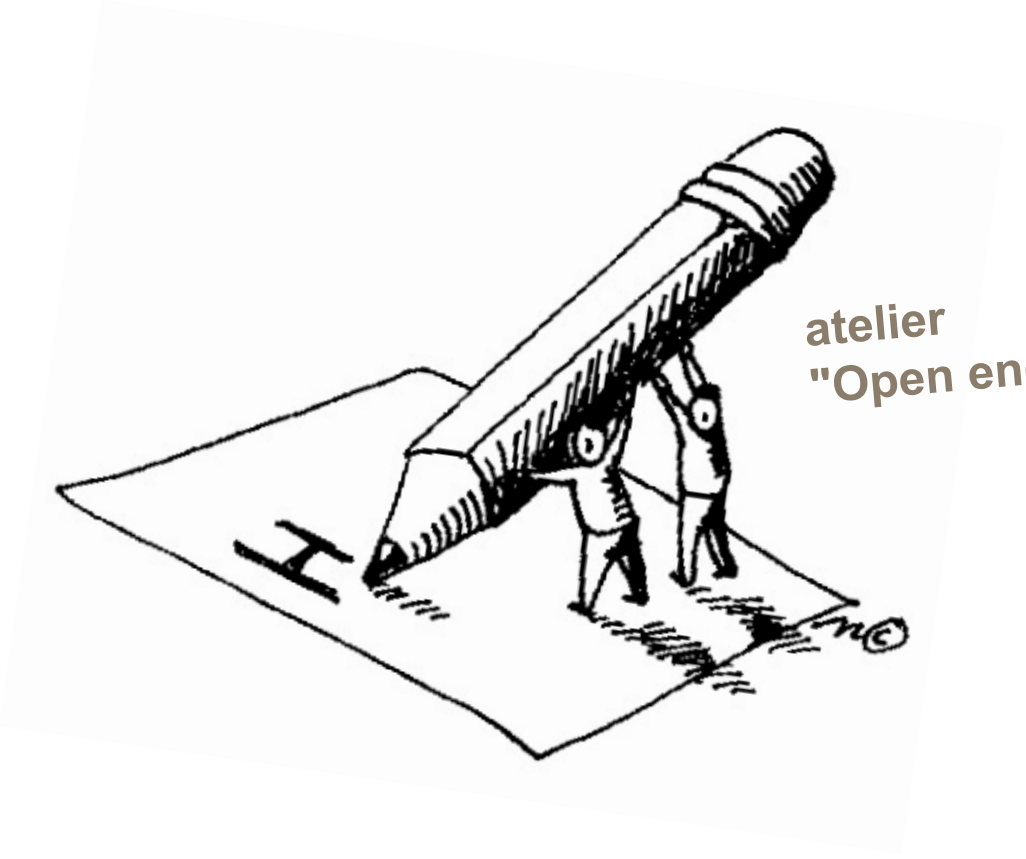
Brainstorming sur une nouvelle version

Définition du besoin



Source : Mountain Goat Software
http://www.mountaingoatsoftware.com/system/presentation/file/119/Cohn-ADP09-Introduction-to-User-Stories.pdf

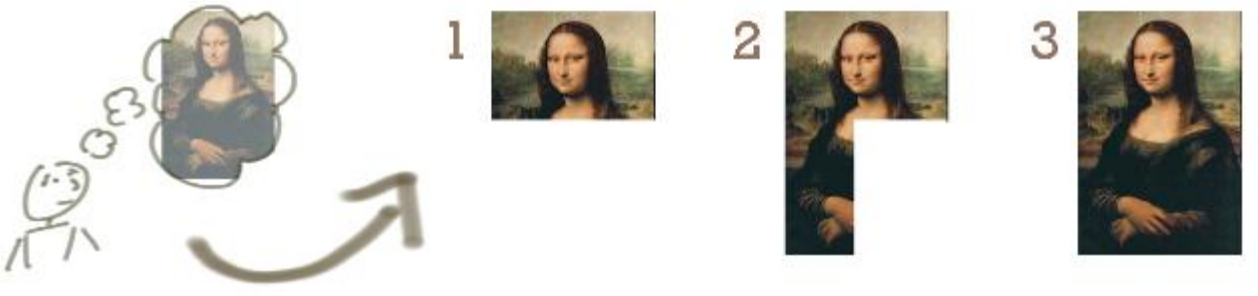
Atelier : "Open Ended Specifications"



atelier
"Open ended specifications"

Source :
The power of open-ended requirements
http://blog.crisp.se/davidbarnholdt/2009/02/18/1234986060000.html

ITERATING VERSUS INCREMENTING



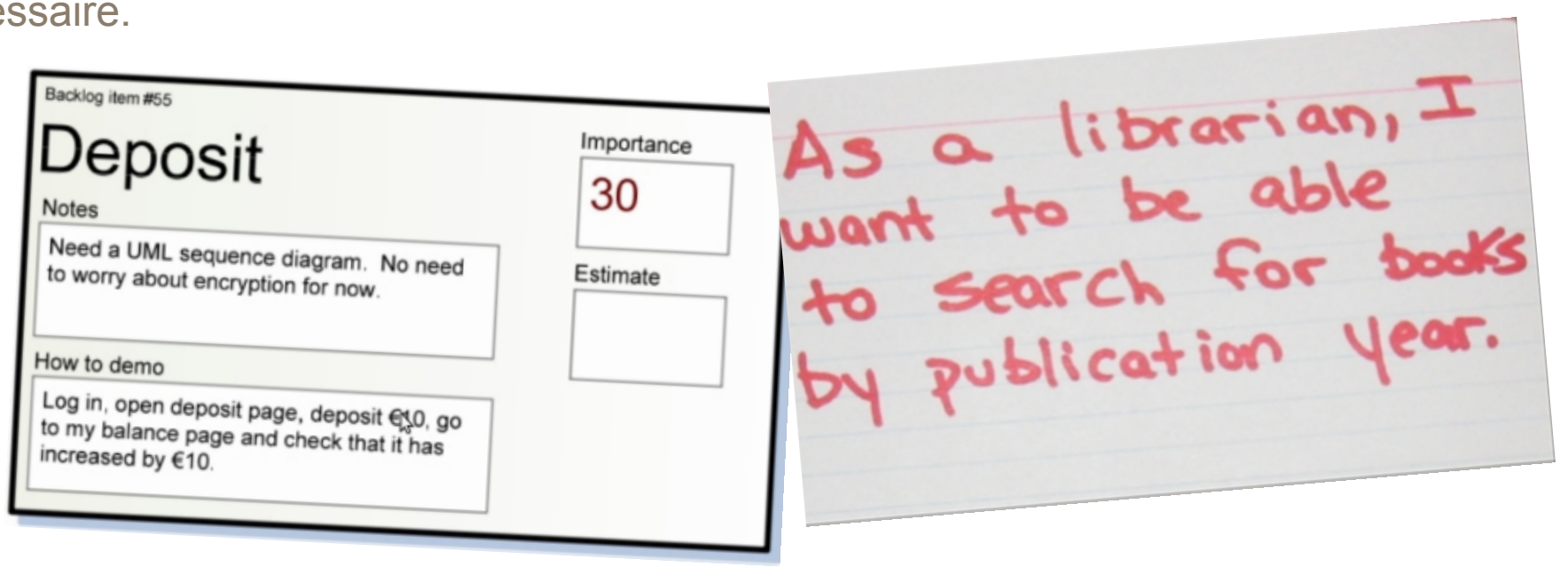
Source : Jeff Patton
http://www.agileproductdesign.com/blog/dont_know_what_i_want.html

User Stories

Les histoires utilisateurs (user story) ne sont pas des cas d'utilisations (use case)

« Une user story (histoire utilisateur) est une exigence logicielle formulée en une ou plusieurs phrases dans le langage de tous les jours ou celui lié au métier de l'utilisateur. Les user stories sont utilisées dans le développement logiciel dit Agile comme spécifications (en même temps que les tests d'acceptance). Le formalisme est limité à un carte de type post-it. » (wikipedia)

Les User Story encouragent à ne pas entrer dans le détail tant que cela n'est pas nécessaire.



User Stories

Ron Jeffries propose une règle des 3C pour gérer les User Story

Card

L'histoire est écrite sur une carte de taille assez réduite.
Ces fiches peuvent être annotées (estimation, etc.)

Conversation

Les détails de l'histoire seront exprimés lors de conversation avec le Product Owner

Confirmation

Des tests d'acceptation sont consignés avec l'histoire pour valider qu'elle a été réalisée correctement.

Source : Ron Jeffries
http://xprogramming.com/articles/expcardconversationconfirmation/

User Stories

Une User Story se base sur la syntaxe suivante :

En tant que <rôle>,

je peux <besoin>

de façon à <bénéfice>

Une bonne user story sera : compacte, testable, estimable, portera de la valeur, négociable, indépendante.

Elle sera comprise par toute la population qui gravite autour et dans le projet

User Stories : L'acronyme "INVEST"

I

Independant (indépendante)

N

Negotiable(négotiable)

V

Valuable (avec de la valeur)

E

Estimable

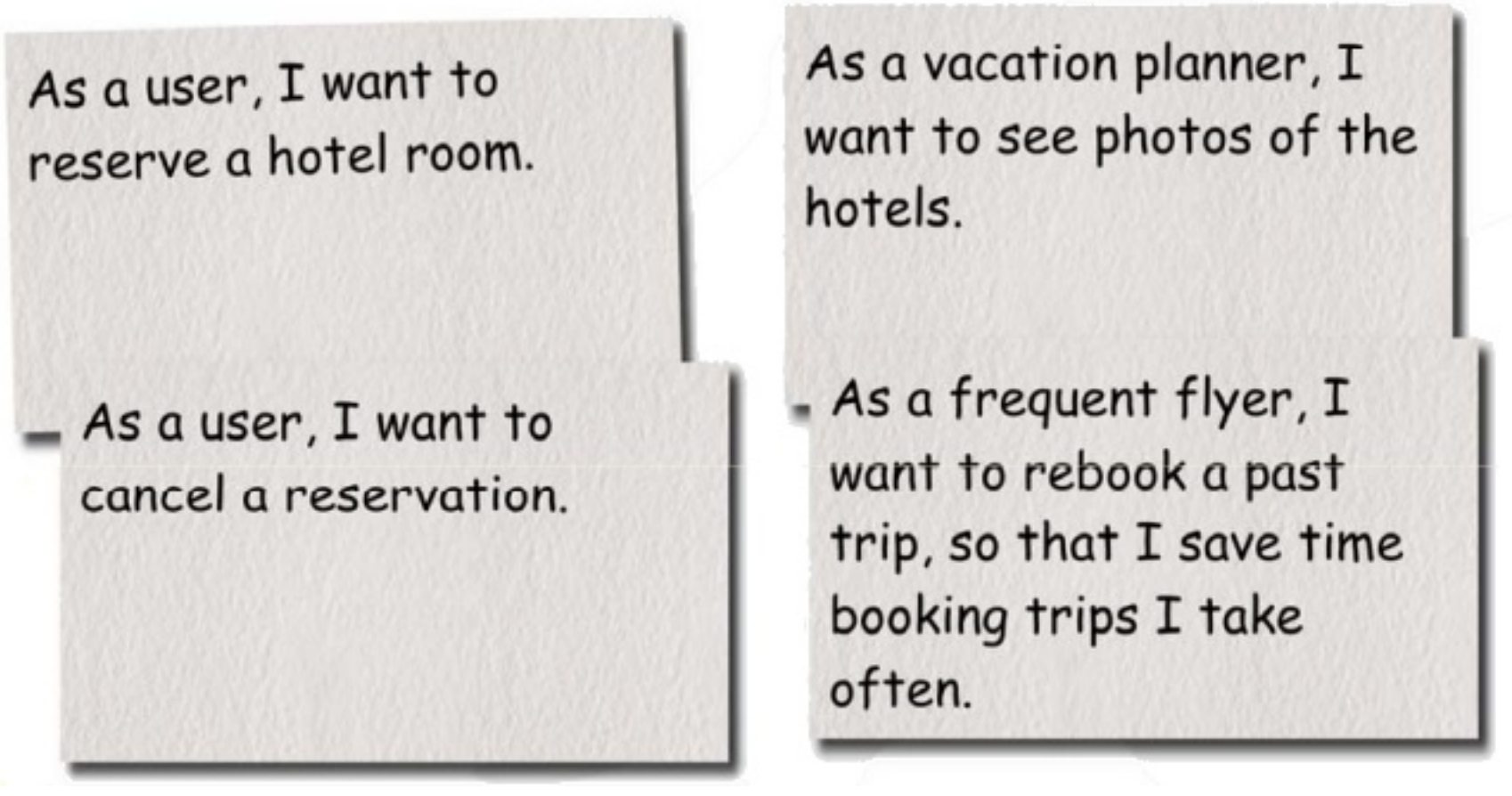
S

Sized to fit (assez petite)

T

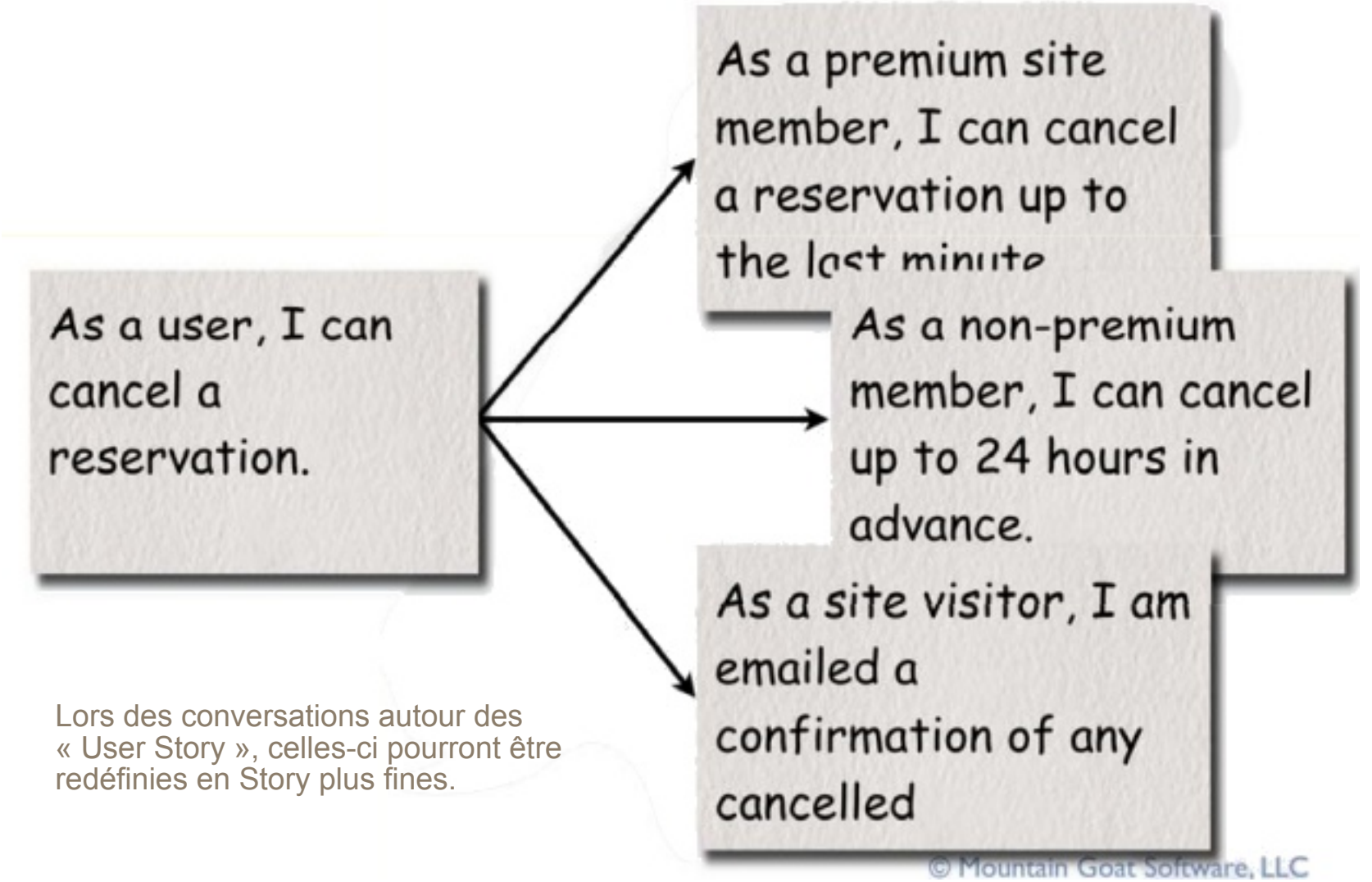
Testable

User Stories



Source : Mountain Goat Software
http://www.mountaingoatsoftware.com/system/presentation/file/119/Cohn-ADP09-Introduction-to-User-Stories.pdf

User Stories



Lors des conversations autour des « User Story », celles-ci pourront être redéfinies en Story plus fines.

Source : Mountain Goat Software
http://www.mountaingoatsoftware.com/system/presentation/file/119/Cohn-ADP09-Introduction-to-User-Stories.pdf

User Stories

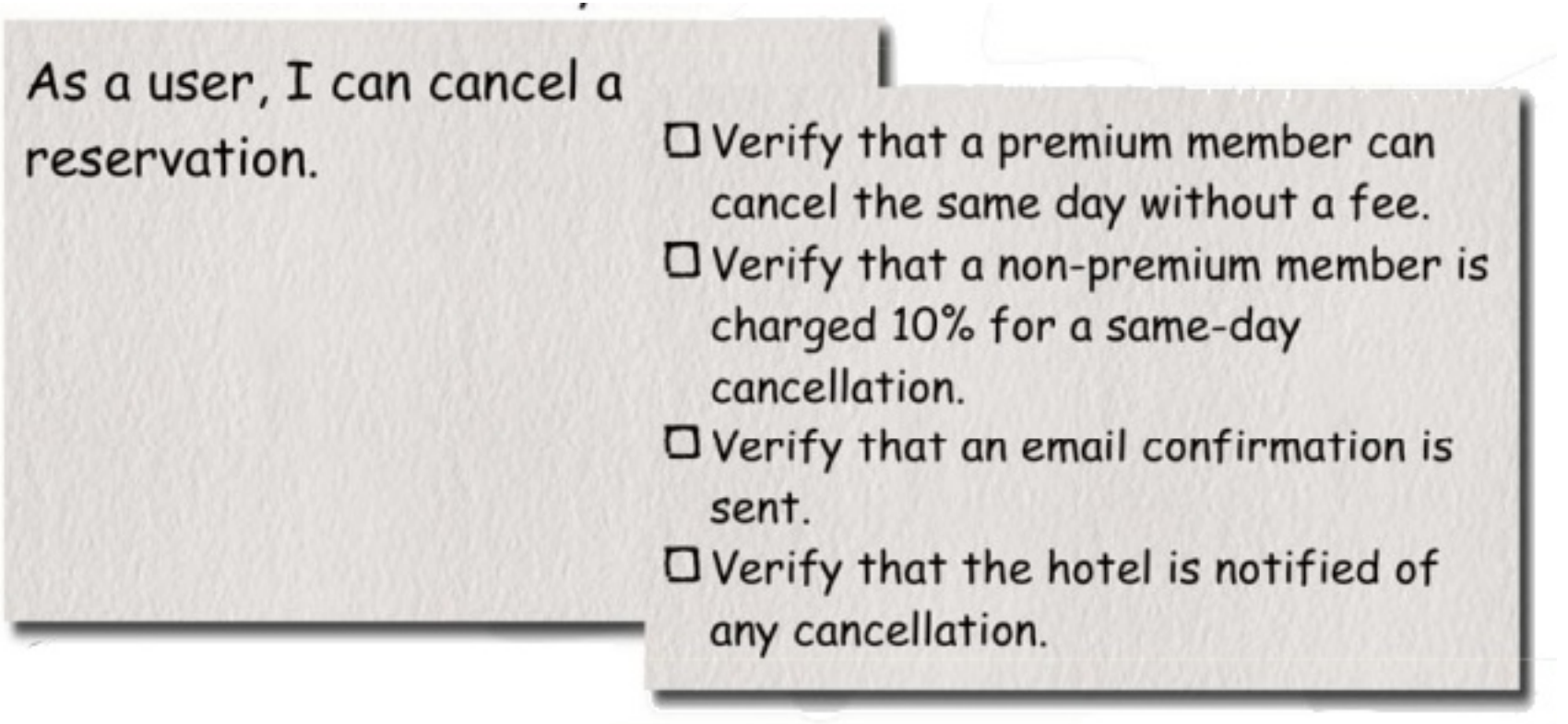
Découper les user stories :

- Workflow steps (par étape)
- Business rule variations (par règle métiers)
- Major effort (par taille d'effort)
- Simple / Complex (approche du simple au complexe)
- Variations in data (variations dans les données)
- Data entry methods (façon d'intégrer les données)
- Defer performance (retarde la question des performances)
- Operations (CRUD) (par opérations)
- Break out a spike (analyser une piste)

Source :
http://www.richardlawrence.info/2009/10/28/patterns-for-splitting-user-stories/

User Stories & Tests d'acceptation

Lors des conversations autour des « User Story » il faudra préciser les tests d'acceptation qui permettront de valider en partie que la Story a bien été réalisée



Source : Mountain Goat Software
http://www.mountaingoatsoftware.com/system/presentation/file/119/Cohn-ADP09-Introduction-to-User-Stories.pdf

Ecrire les ATDD avec du Gherkin

Le Gherkin est un langage très simple pour formaliser et unifier l'écriture de vos tests d'acceptation. D'autant qu'il pourra à terme être utilisé par des outils qui vont permettre d'automatiser ces tests, ou du moins fournir facilement une enveloppe de tests (specflow, cucumber, jbehave, etc.)

Etant donné la page d'accueil du site avec un bouton « enregistrez vous »
Quand je clique sur le bouton « enregistrez vous »
Alors j'arrive sur le formulaire d'enregistrement

Etant donné le formulaire d'enregistrement
Quand je m'enregistre
Alors je reçois un email de demande de confirmation
Et cet email contient un lien de confirmation

Etant donné un lien dynamique sur une page de confirmation d'enregistrement
Quand j'accède à la page mon enregistrement est confirmé
Alors je reçois un email de confirmation d'enregistrement
Et je peux désormais me connecter au site

User Stories & Personas

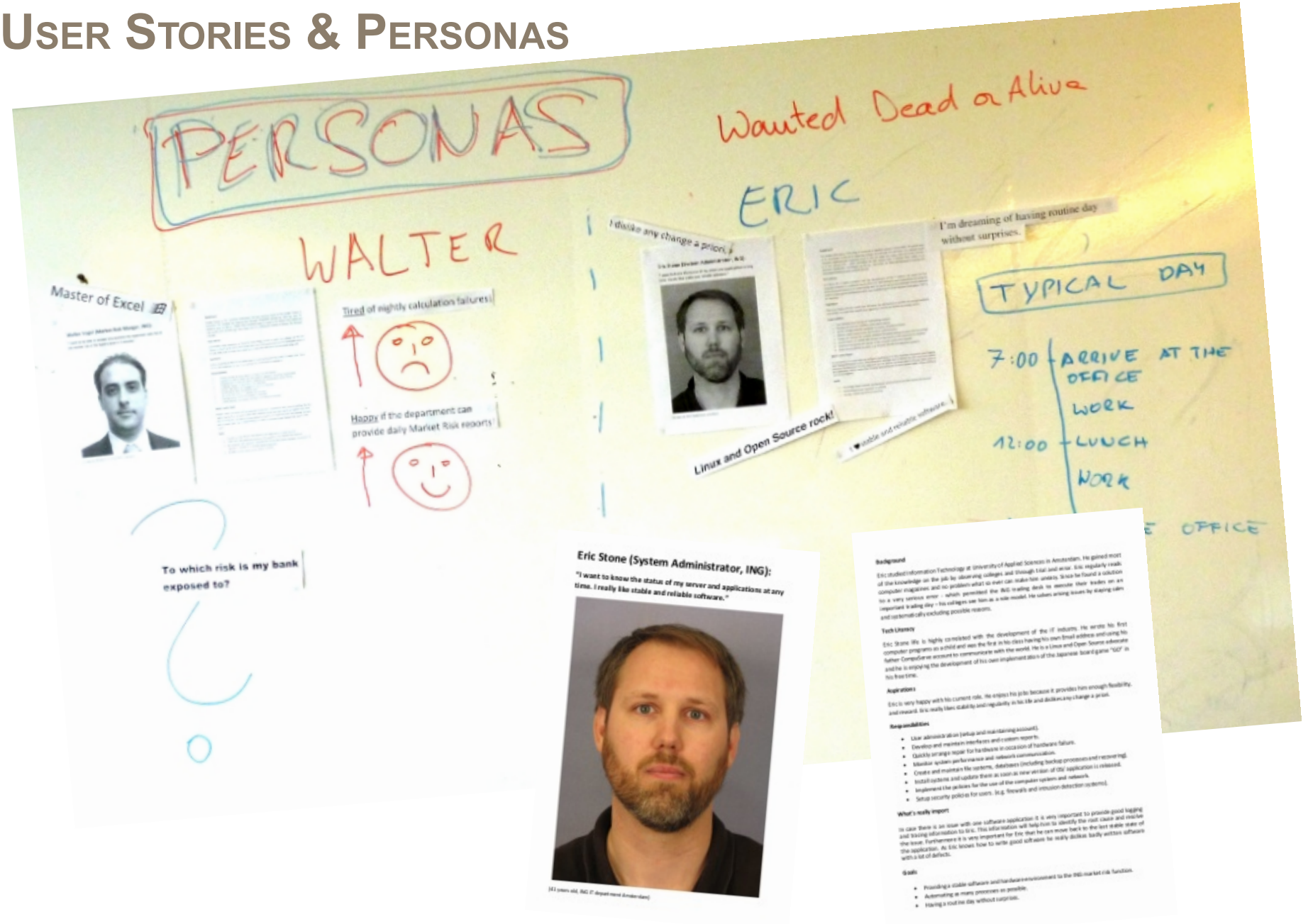
Il est souvent bienvenu de définir et de discuter des rôles (on dit des "personas" / personnages) du projet.

Certains poussent le « vice » à les nommer, fabriquer des photos, décrire leur caractère.

L'utilisateur devient réel : on commence à concevoir le produit comme une réponse à un vrai besoin, à une vraie nécessité.

Pour cela ne dites donc plus systématiquement « l'utilisateur » mais le « responsable des ventes », « la personne qui voyage beaucoup », etc.

User Stories & Personas

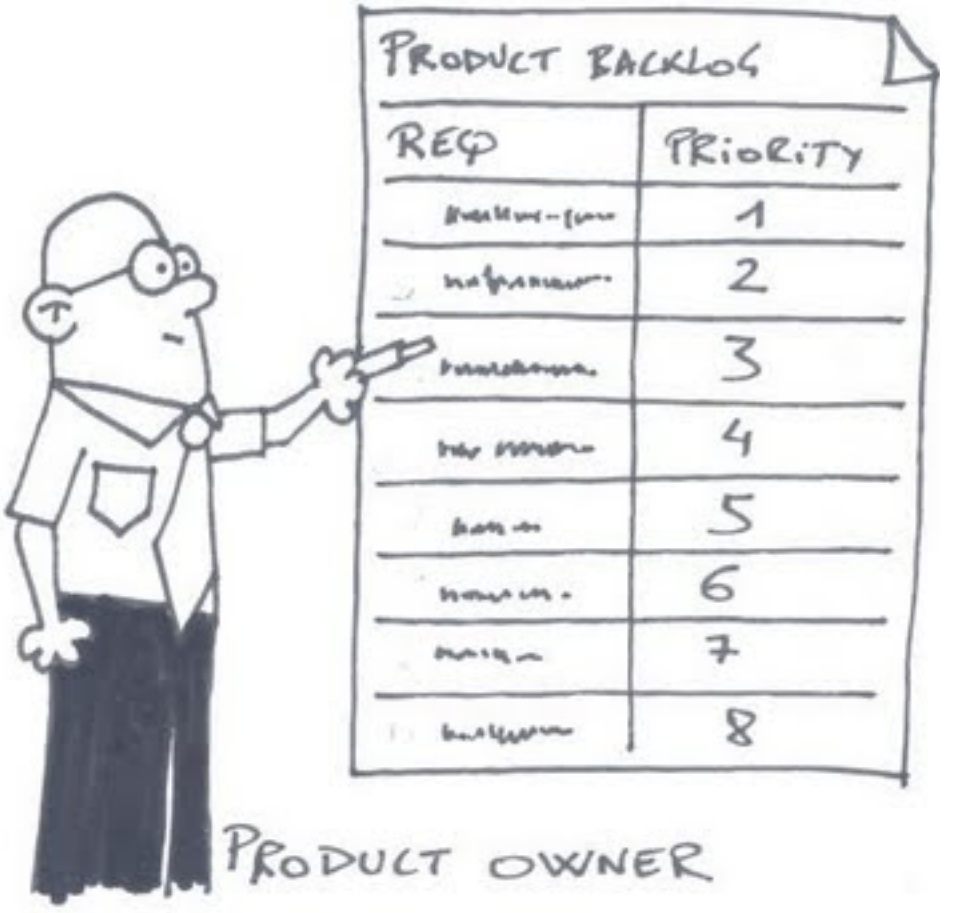


Définition du besoin : le Product Backlog

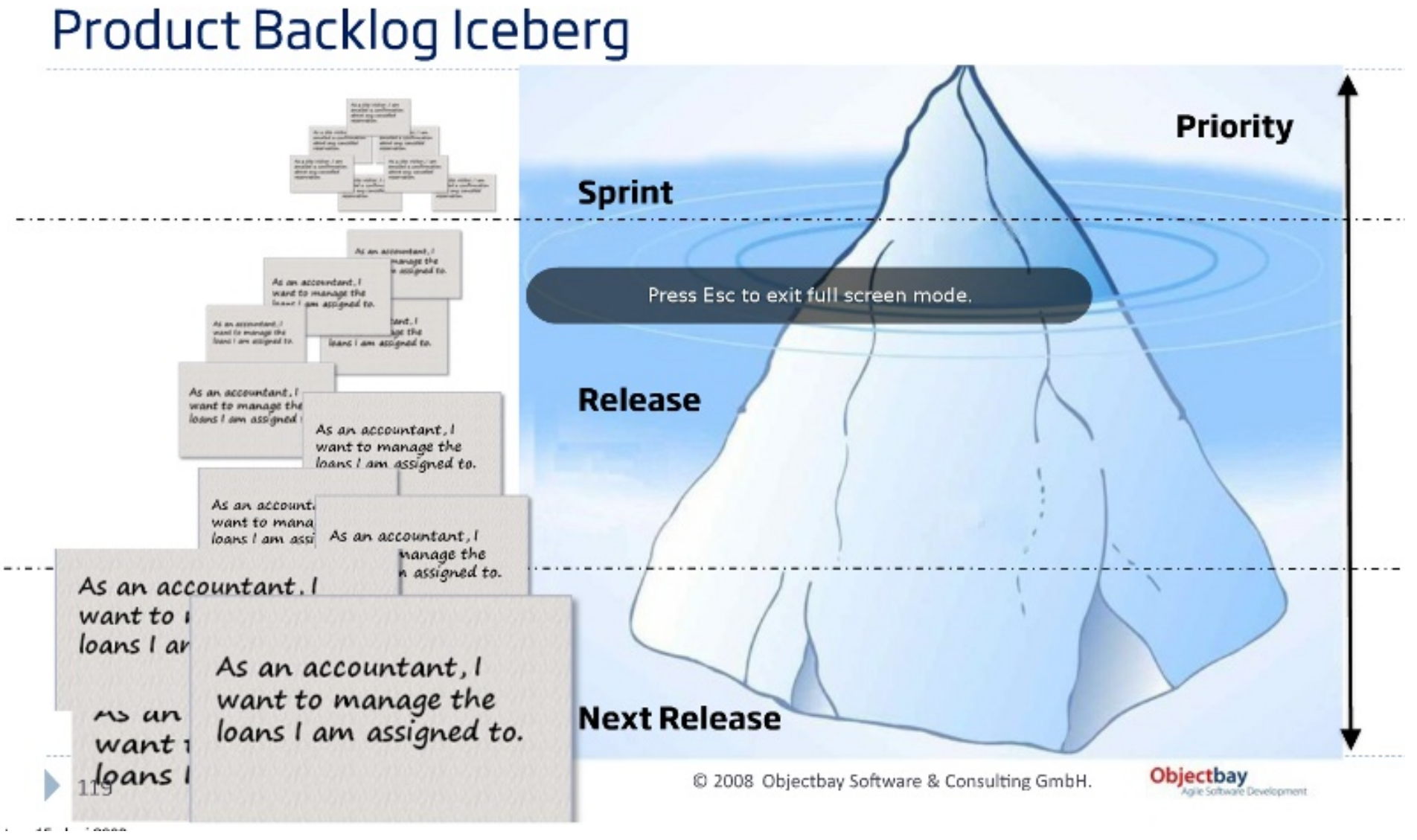
La liste de tous les besoins exprimés que le Projet a pour cible

Une rencontre entre les besoins métiers et le point de vue technique

Exprimé de façon à mettre en évidence la valeur apportée par chaque composant (User Story)

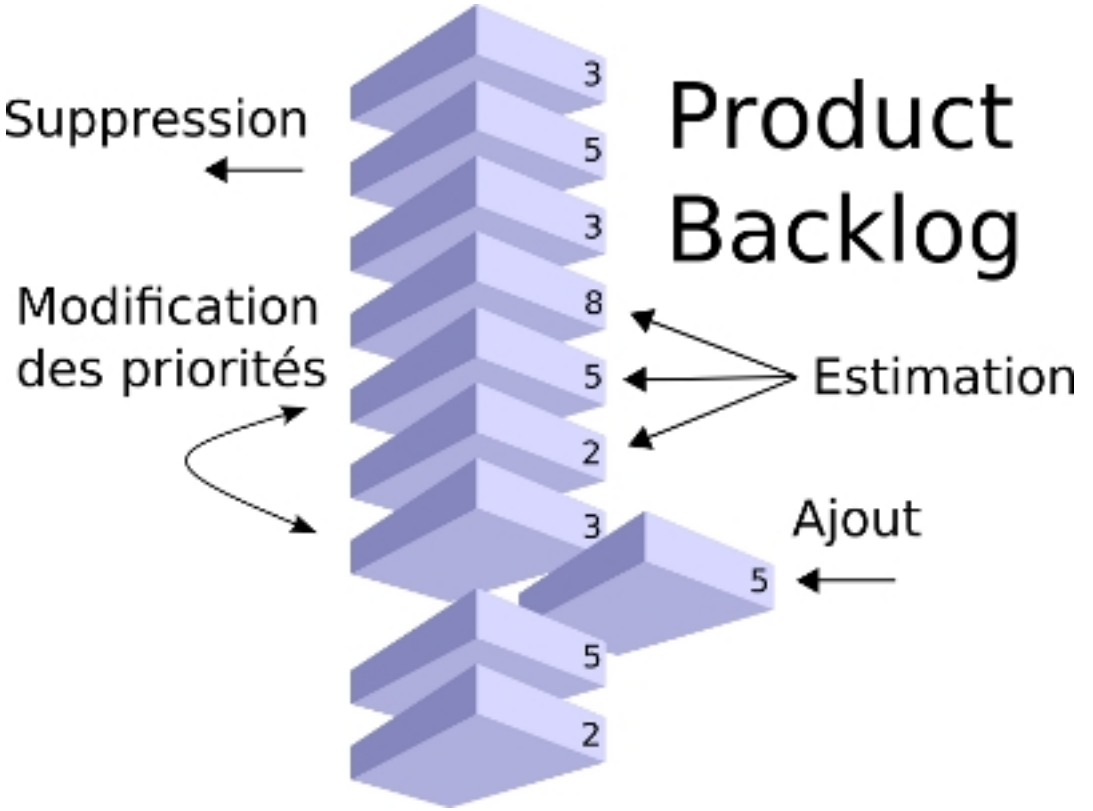


Product backlog Iceberg



Product backlog

Priorisé par le Product Owner, repriorisé en début de chaque sprint, ou à tout moment



Atelier : Définir un backlog

As a	I want to	So that (I can)	Business Value	Estimate
HR Manager	Publish new vacancies	Find candidates	80	20
Job Hunter	Apply for a job	Quickly apply for a job	80	40
HR Manager	Triage applicants	Politely eliminate unpromising candidates	50	8
Googlebot	effectively find and index all postings	Ensure that internet searchers can find job postings on this site	50	13
System Admin	quickly recognize and analyze system	ensure rapid resolution of technical problems	30	20

atelier
Backlog

Notion de "fini" / Done

Cette notion a pour objectif de fixer le niveau de qualité attendu, et ce dans tous les cas de figure. C'est une condition pour dire qu'une user story est finie.

Cette notion doit être partagée et connue de tous (au mieux on l'affiche sur les radiateurs d'informations !)

Généralement elle est partagée (au moins un tronc commun) entre les différents projets, au niveau de l'entreprise

Cette notion est très liée à la notion de test (unitaire et d'acceptation) mais il faut la traiter de façon verticale : pas uniquement le côté code, ou architecture, ou ergonomique, etc.

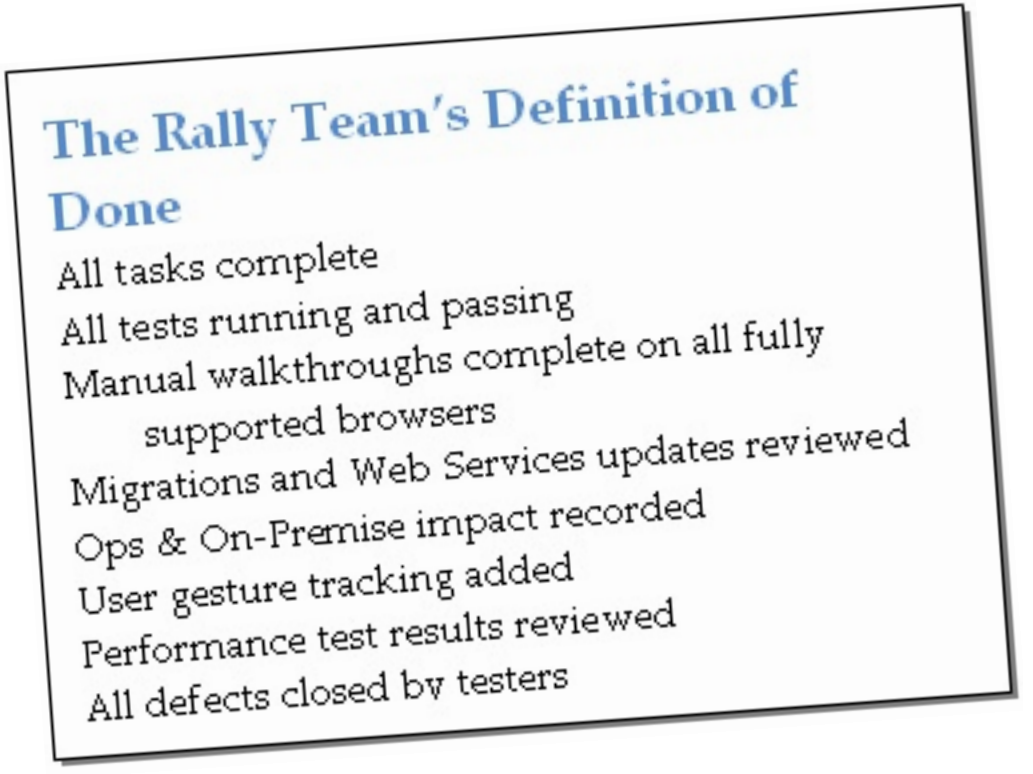
La notion de « done » implique plusieurs couches : technique, métiers, IHM, documentation, etc

Notion de "fini" / Done

Définissez les rôles d'un projet

Quelle serait la notion de « fini » pour chacun de ces rôles ?

Quelle serait la notion de « fini » sur le projet Scrum de cette équipe ?



Pratiques du dev. itératif : les itérations ou sprints

On parlera de Sprint ou d'itération

Il dure entre 2 semaines ou 1 mois selon les équipes, mais sa durée ne varie plus dans un même projet.

Une durée de 2 semaines facilite la prise en main de Scrum car elle donne de la visibilité à intervalle resserré.

Il faut décomposer les tâches de façon à ce qu'elles durent un jour maximum pour assurer de la visibilité sur l'avancement

Si en cours de Sprint l'équipe souhaite changer la priorité (sortir une story, en intégrer une nouvelle en provenance du Product Backlog) c'est toujours le Product Owner qui aura le dernier mot

Concernant le Sprint Backlog c'est l'équipe qui s'auto-organise

Le sprint planning

Le Sprint Planning permet de définir le Sprint Backlog

Le Sprint Backlog est un sous ensemble du Product Backlog

Le Sprint Backlog correspond à un accord mutuel entre le Product Owner et l'équipe mais c'est le Product Owner qui a le dernier mot.

Le Sprint Planning se découpe en deux étapes

Définition du contenu du Sprint
(quoi ?)

Estimation et découpage en tâches du contenu du Sprint
(comment ?)

Le sprint planning : quoi ?

En fonction de la dernière revue de sprint, de la dernière rétrospective, d'autres informations, le ScrumMaster aura pris soin de sensibiliser le Product Owner à valider ou à re-prioriser le Product Backlog avant cette réunion

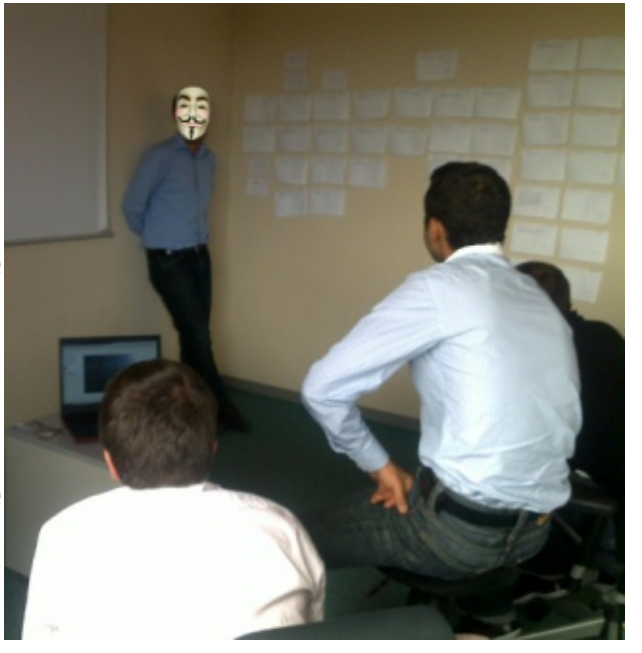
Le Product Owner présente le Product Backlog à l'équipe

L'équipe estime quelles « user story » (dans l'ordre des priorités) elle est susceptible d'intégrer (la vélocité est une indication forte). Elle s'engagera là dessus

L'équipe peut proposer et justifier un ordonnancement du Product Backlog différent mais c'est le Product Owner qui aura le dernier mot.

On peut dans cette phase estimer les User Story qui ne le sont pas encore (on va travailler généralement sur 60/70% du Product Backlog)

On n'assigne jamais une User Story à l'équipe, c'est l'équipe qui définit son propre potentiel et qui s'auto-organisera quand à l'affectation des tâches



Le sprint planning : quoi ?

Objectif

Cette première partie de « Sprint Planning » est aussi le moment où l'on défini le but du Sprint, son Leitmotiv, son objectif, sa cible. Par exemple : « Mise en œuvre du système de réservation en ligne des croisières » ou « Réalisation des échanges entre le site web et l'ERP concernant les réservations en lignes »

Cet objectif doit être l'élément indispensable à atteindre pour que l'on puisse considérer que le sprint est une réussite.

Si des arbitrages doivent avoir lieu durant le Sprint, c'est souvent l'objectif du print qui permettra de faire pencher la balance.

Engagement

La fin de cette première partie de Sprint Planning se conclue avec l'engagement de l'équipe sur un périmètre.

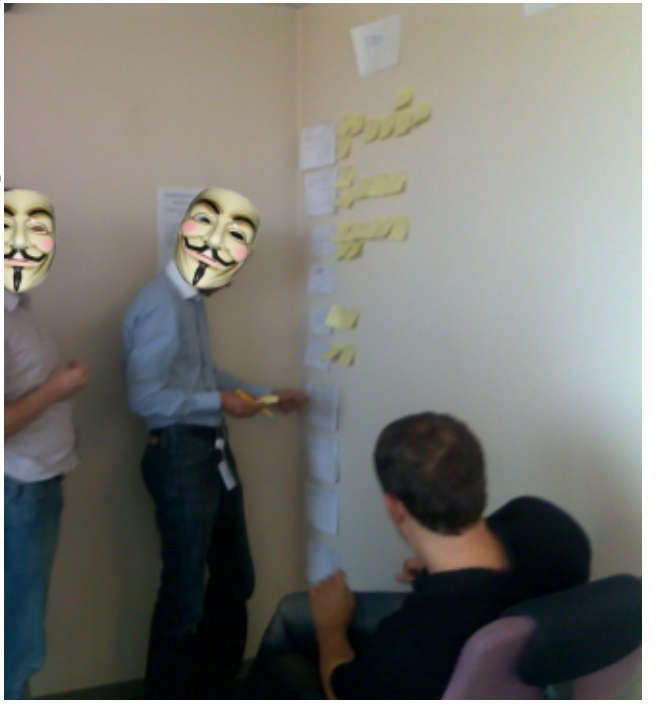
Le sprint planning : comment ?

La seconde partie du Sprint Planning consiste à découper les User Story en tâches.

Il est nécessaire de discuter du contenu de chaque Story et de l'intérêt de chaque tâche

Il ne faut pas oublier les tâches de spécifications, de documentation, de test, etc. Tout ce qui est nécessaire pour la Story soit achevée convenablement (voir la notion de « done » (fini)).

On peut ajouter des tâches sans Story (bugs, etc.) mais essayez plutôt d'ajouter des Story de type « process » ou « system » (afin que la vélocité reste au plus juste)

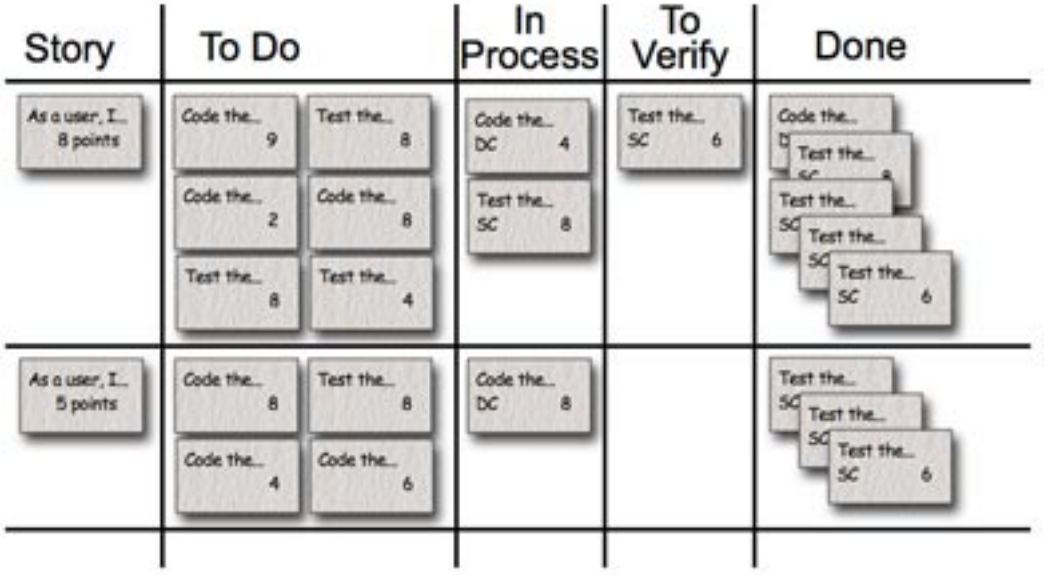


Le sprint planning : comment ?

Les tâches sont estimées (ou pas) en heures selon la maturité de l'équipe (on peut se passer de l'estimation).

On peut aussi estimer les tâches en point (avec un nouveau planning poker, mais c'est relativement rare et très chronophage)

L'estimation d'une tâche en heure ne devrait pas dépasser 1 jour de façon à pouvoir rapidement détecter les dérives le cas échéant



Itération : fin exceptionnelle ? de stabilisation ? Sprint 0 ?

Le Sprint peut-être annulé

Il n'a plus de valeur !

Il est impossible de le réussir

Seul le Product Owner peut annoncer l'annulation et la fin du Sprint (mais l'équipe ou le management peuvent essayer de le convaincre)

Les « user stories » estimées « finies » sont auditées

Tous les « user stories » non achevées sont replacées dans le Product Backlog

Existe des « sprint de stabilisation » ?

Qu'est ce qu'un Sprint 0 ?

Pratiques d'estimation

L'estimation est collective, c'est l'équipe qui la réalise mais en conversant avec le Product Owner et le Scrummaster

L'estimation est exprimée en « story point » à partir d'un planning poker, soit exprimés en « jour idéaux ». Je préconise le planning poker car toute notion de durée (jour) vient troubler la réflexion. Des fois on utilise même des « points animaux » ou des tailles de tee-shirt...

L'estimation est relative : une « story » par rapport aux autres (triangulation)

Attention de bien prendre en compte tout ce qui a été prévu par « done »

En fonction de l'estimation réalisée sur des story il se peut que le Product Owner re-priorise son Product Backlog

Estimation de la complexité, de la durée des tâches à accomplir, de la prise de risque, etc.

Pratiques d'estimation

Point de Story

- Mesure un effort
- Relatifs jamais absolus
- Difficile à faire comprendre au management

Jours idéaux

- Mesure une durée
- Devraient être relatifs
- Plus aisés à expliquer aux personnes externes
- On les raccroche trop à une notion de calendrier

les jours ideaux : à ne pas utiliser ! juste un rappel historique.

Le "Planning Poker"

Chaque personne possède un jeu de planning poker

Le Product Owner décrit la Story devant être estimée

On discute de la story

Tout le monde propose sa mise simultanément

Si toutes les estimations convergent, c'est ok

Si les estimations ne convergent pas, les discussions reprennent, notamment entre les deux personnes ayant les plus grands écarts de point.

Le "Planning Poker"

Ca marche

Car ceux sont les gens qui réaliseront la tâche qui l'estiment !

Le planning poker nécessite que l'on justifie les points que l'on attribue

Propose un étalonnage assez réduit qui limite les erreurs (si une Story est trop grosse, on la découpe en plusieurs Story)

Les valeurs sont relatives

Les valeurs sont pré-définies (on ne chipote pas...)

Tout le monde a son mot à dire (parmi l'équipe)

C'est rigolo

Une alternative le "Wall Planning Poker"

On positionne les stories par comparaison en colonne sur un mur

On affecte les points ensuite

C'est très rapide et on oubli toute valeur numérique !

Mais l'investissement n'est pas forcément égal entre tous les participants

Ma recommandation : faire des wall planning poker hors des sprints planning pour faire des estimations assez loin dans le temps ou initier un projet, utiliser le planning poker lors des sprint planning.

Atelier : Planning poker et Wall Planning poker

ateliers
Planning Poker
Wall Planning Poker

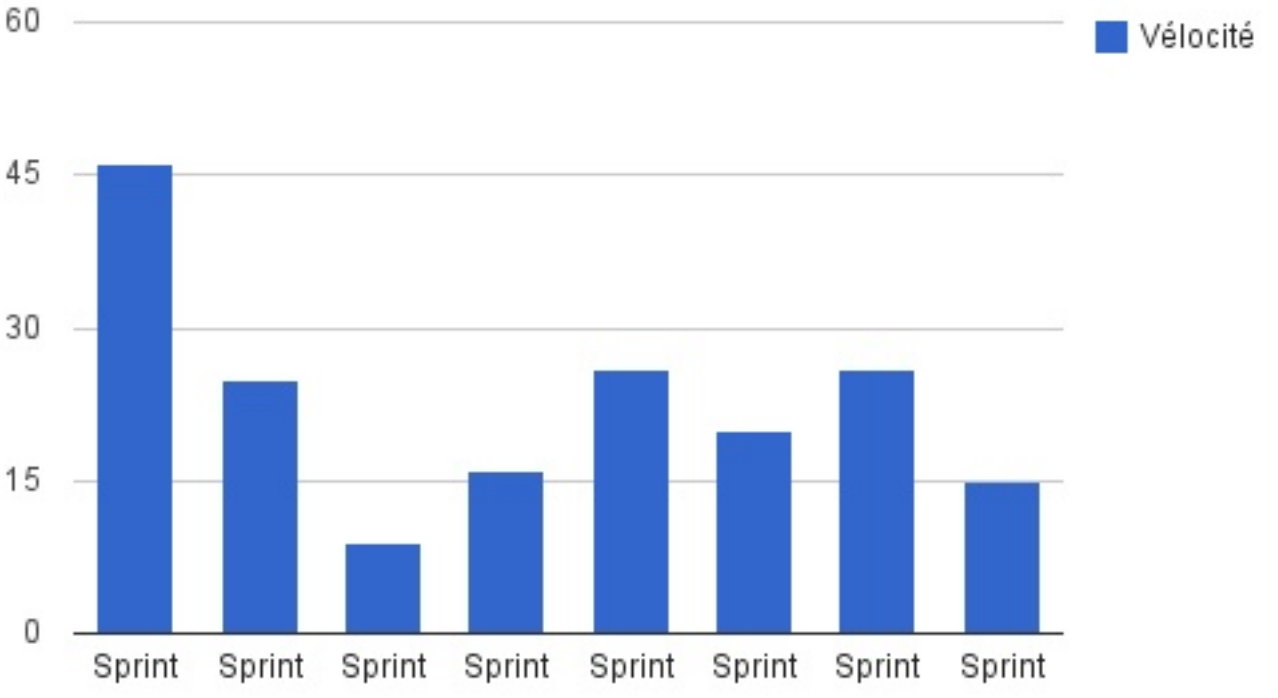
Pratique de planification : la "Vélocité"

Est calculée à la fin du sprint (itération)

C'est la somme des points des « Story » qui ont été acceptées pour la démo

La régularité de la courbe de la vélocity est signe du respect de la qualité, ou de la stabilité de l'environnement (et la stabilité de l'environnement mène au respect de la qualité)

Si on décide de rompre avec la qualité pour améliorer la vélocité (fin des tests, etc.) on va améliorer celle-ci temporairement puis décliner vers un « noyau foutu ». C'est la notion de dette technique.



Planification de releases

Une version de produit à une « release » et à un product backlog

Une « release » contient plusieurs sprints (itérations)

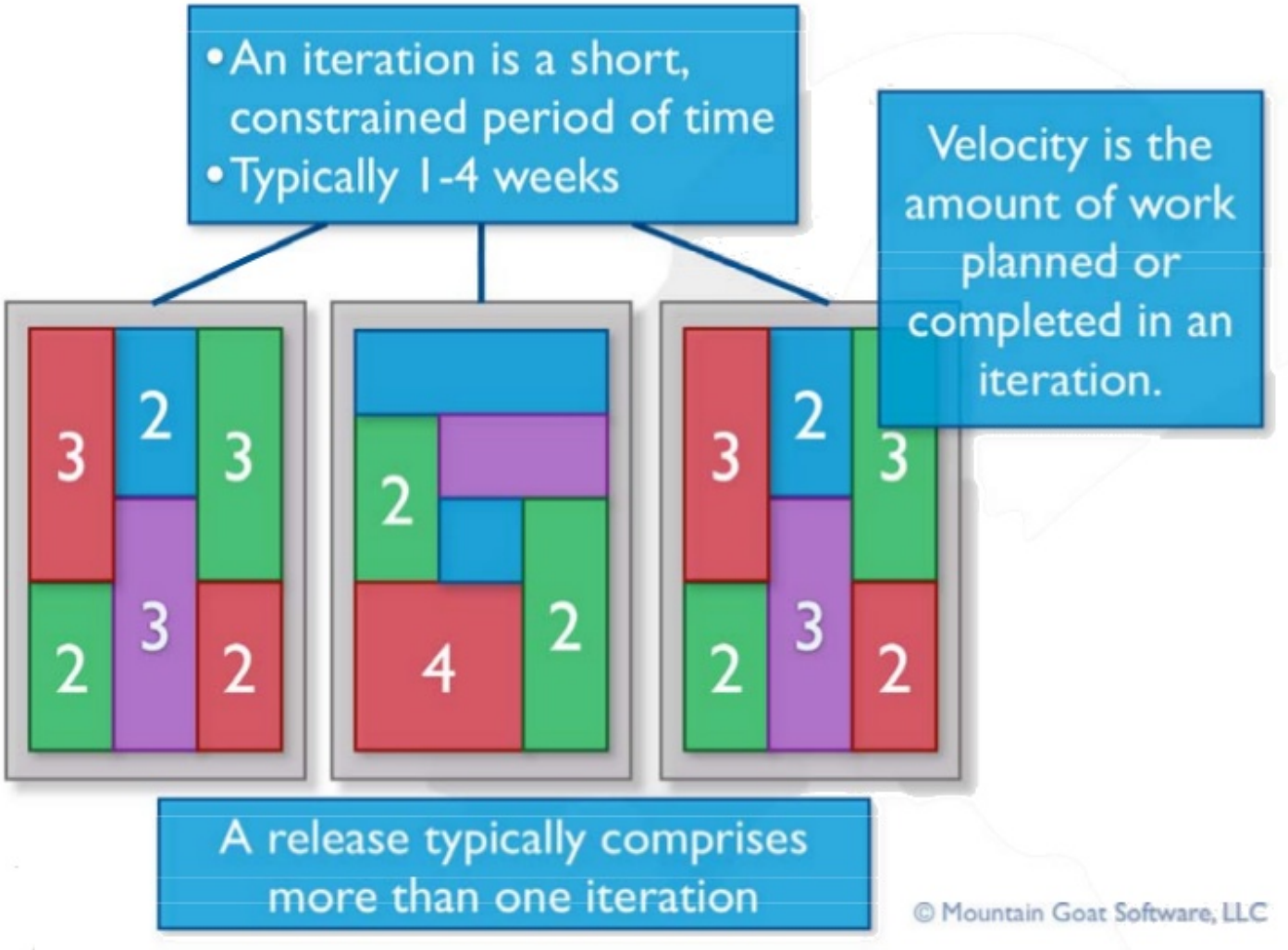
Chaque sprint possède une vélocité

La vélocité est une valeur (non relative à une durée comme jour ou heure) que l'on déduit en additionnant la vélocité de chaque story (nous verrons comment nous calculons la vélocité plus loin)

Chaque sprint est « time-boxé » (disons 3 semaines)

L'équipe est sensée être pérenne et donc la vélocité devrait donc peu ou proue rester la même par sprint

PLANIFICATION DE RELEASES



Source : Mountain Goat Software
http://www.mountaingoatsoftware.com/system/presentation/file/51/bayXP_070320_PlanningAgileProjects.pdf

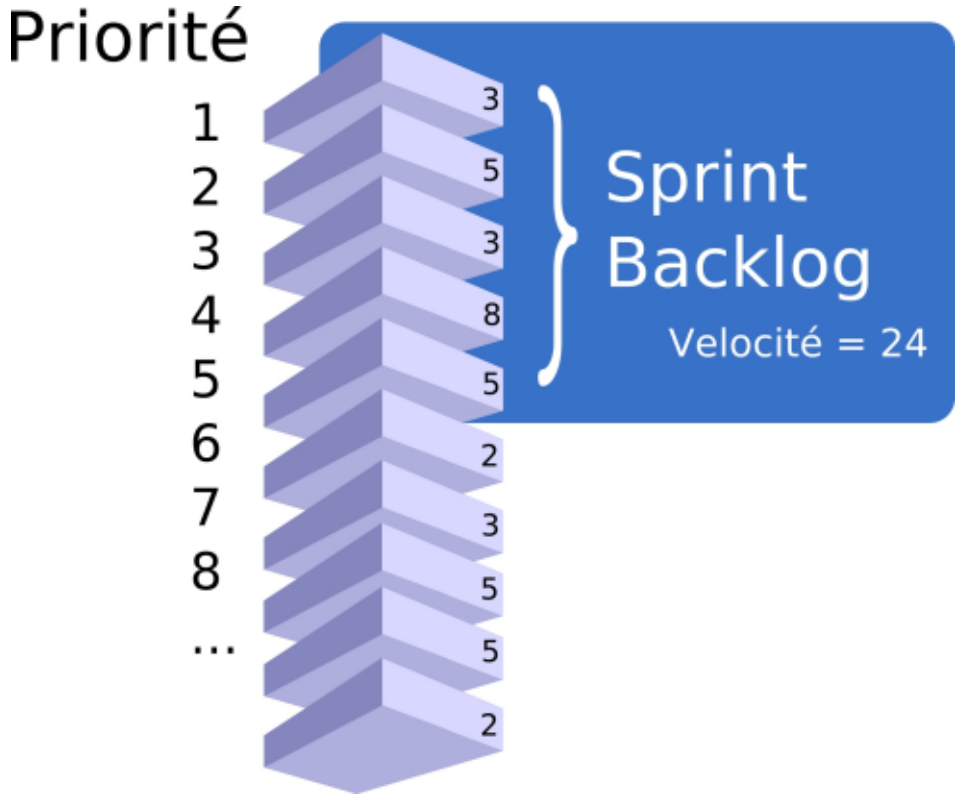
Planification de releases

On sait donc calculer le nombre de sprints nécessaires pour atteindre la fin de la release.

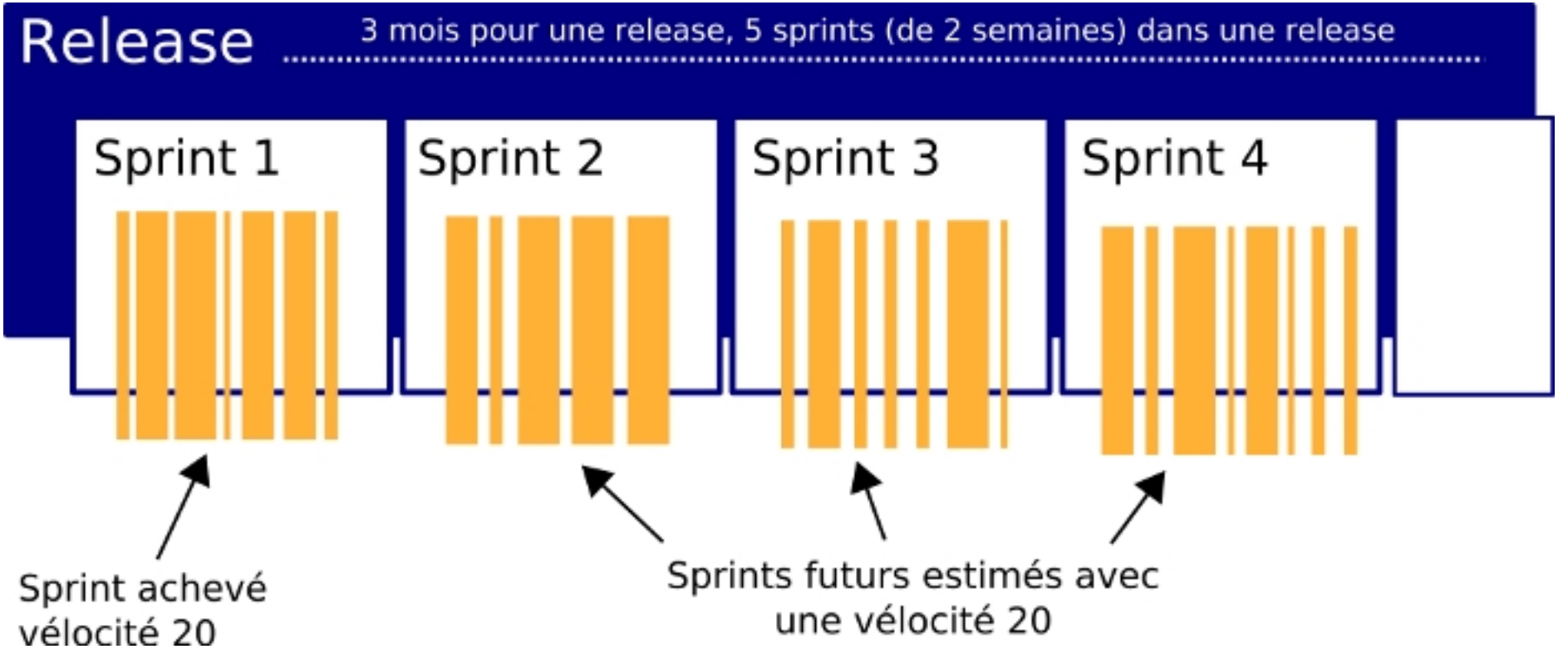
La fin de la release cela peut être

- Le Product Backlog est vide (rarement le cas...)
- Une date prédéfinie
- La fin du budget
- Assez de valeur produite
- Etc.

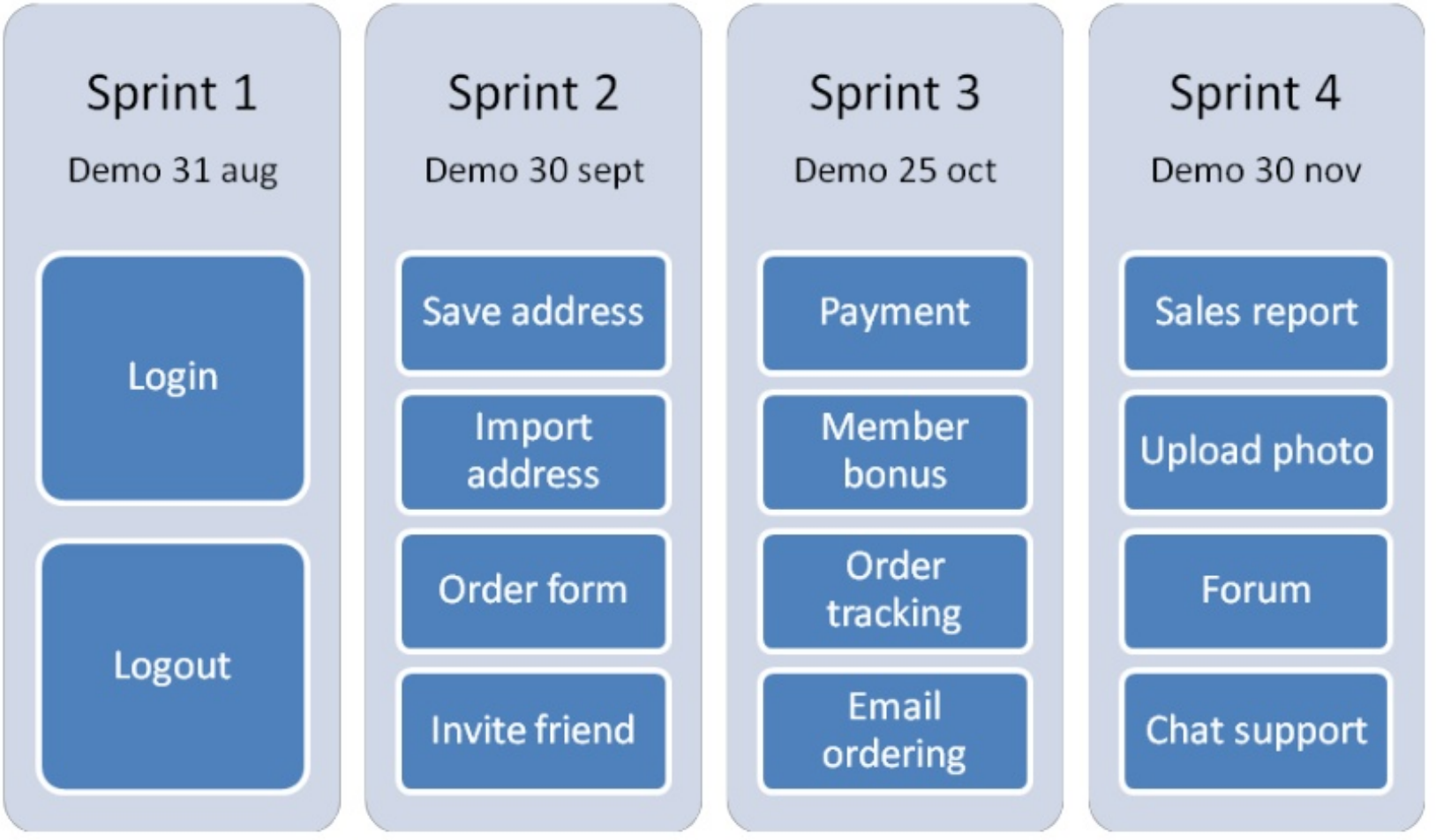
Planification de releases



Planification de releases

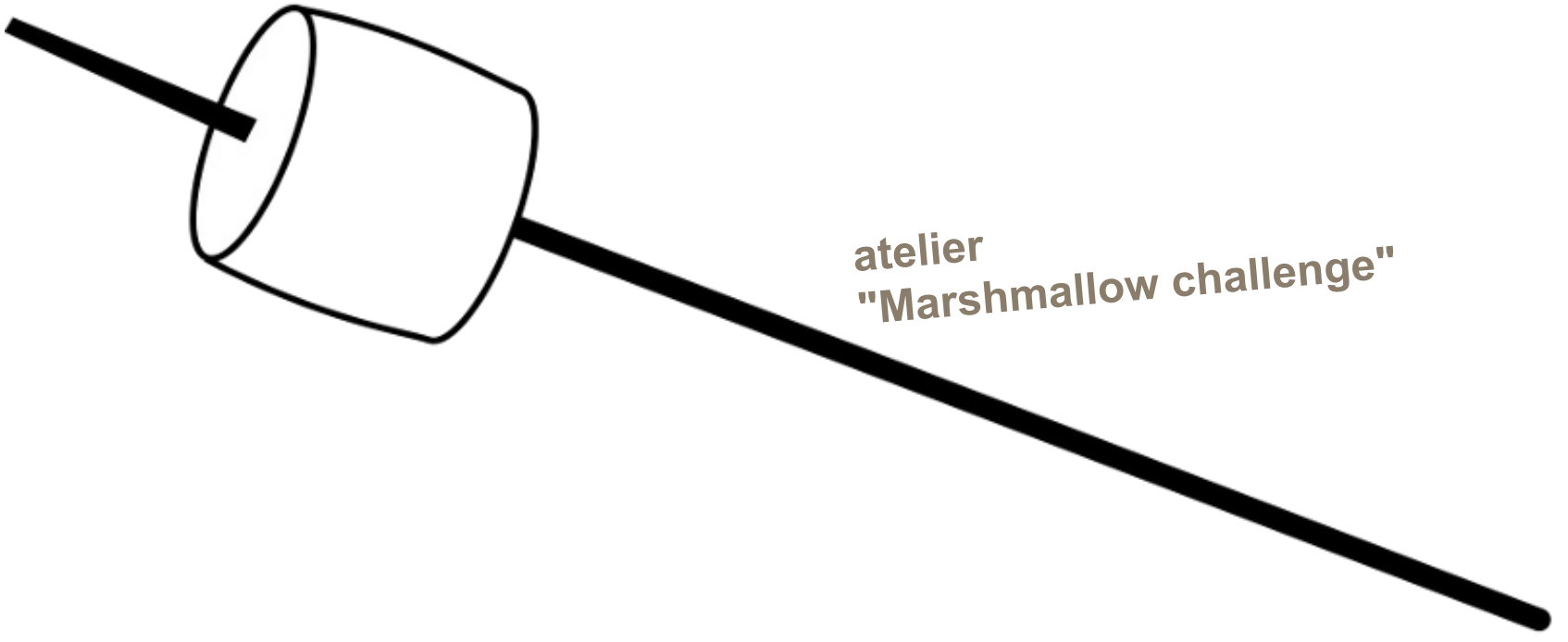


Release planification



Source :
http://blog.crisp.se/mattiasskarin/tags/teams/

Atelier : "Marshmallow Challenge"



http://marshmallowchallenge.com/Welcome.html

Radiateurs d'information

Ils permettent de suivre de façon collective, interactive, et simple l'avancée du sprint, et du projet

Ils engagent et motivent l'équipe

La mise à jour est quasi en temps réel

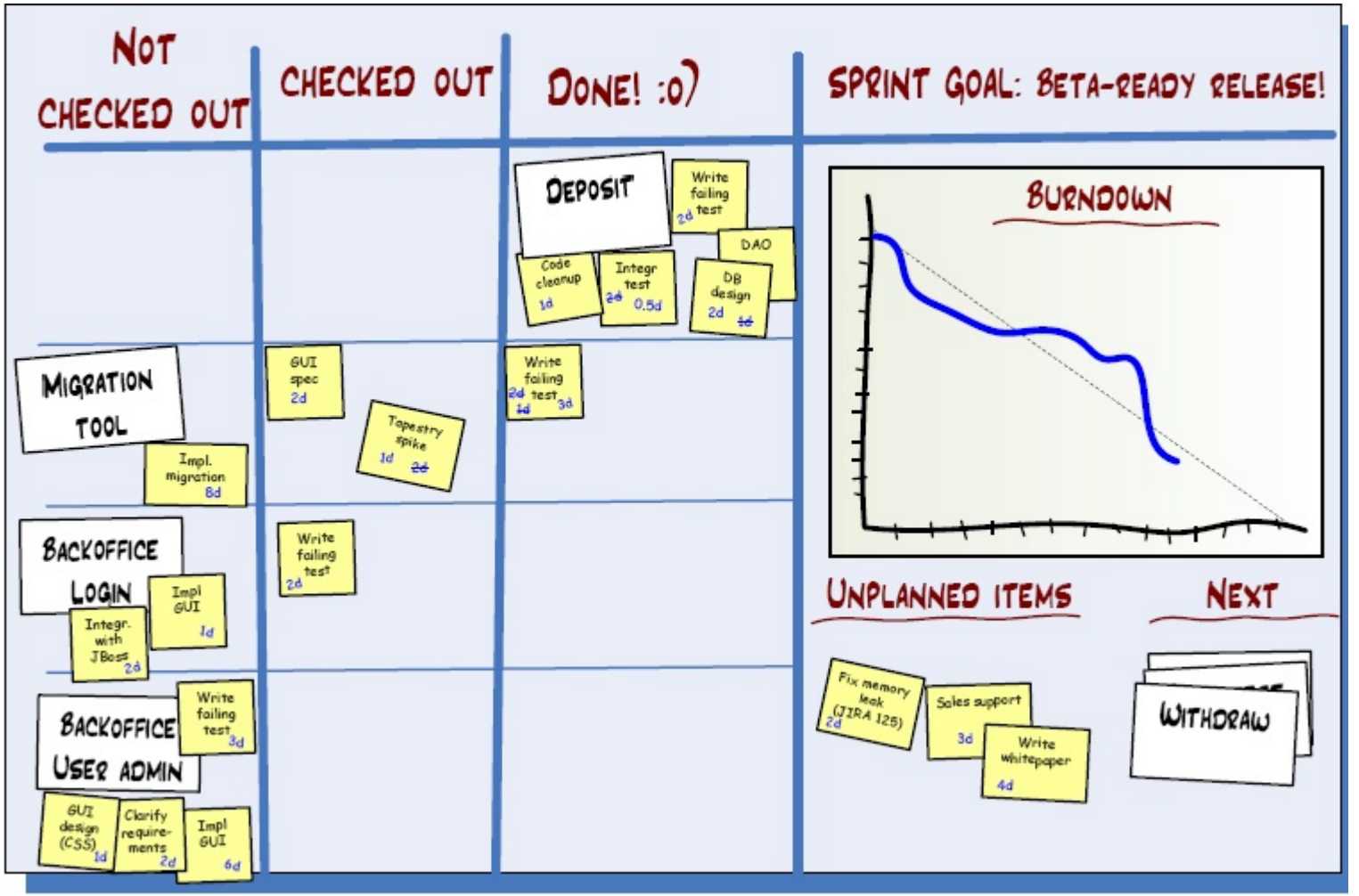
Ils doivent être visibles par tous (autant par l'équipe que par les parties prenantes).

Toutes les informations clefs sont rassemblées en un seul lieu

Ps : attention aux coups de vent, aux ménages le soir, aux post-it défectueux que l'on retrouve au sol...

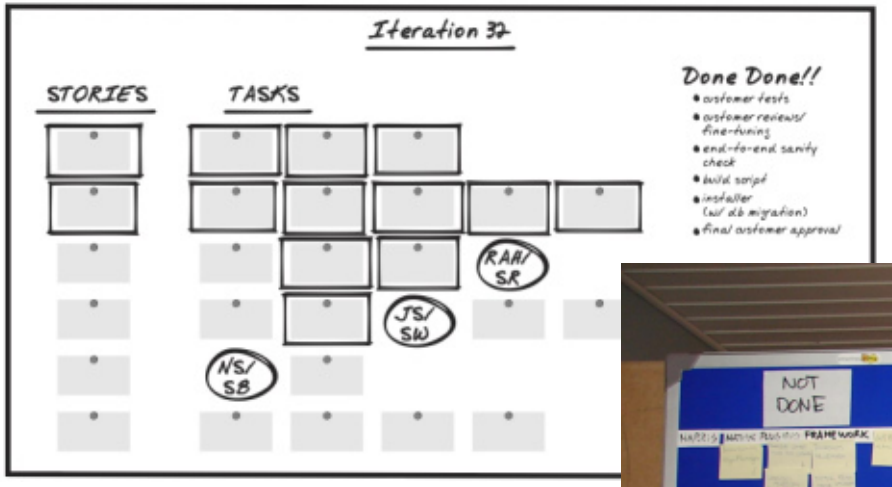
Astuce : prendre en photo le radiateur au jour le jour

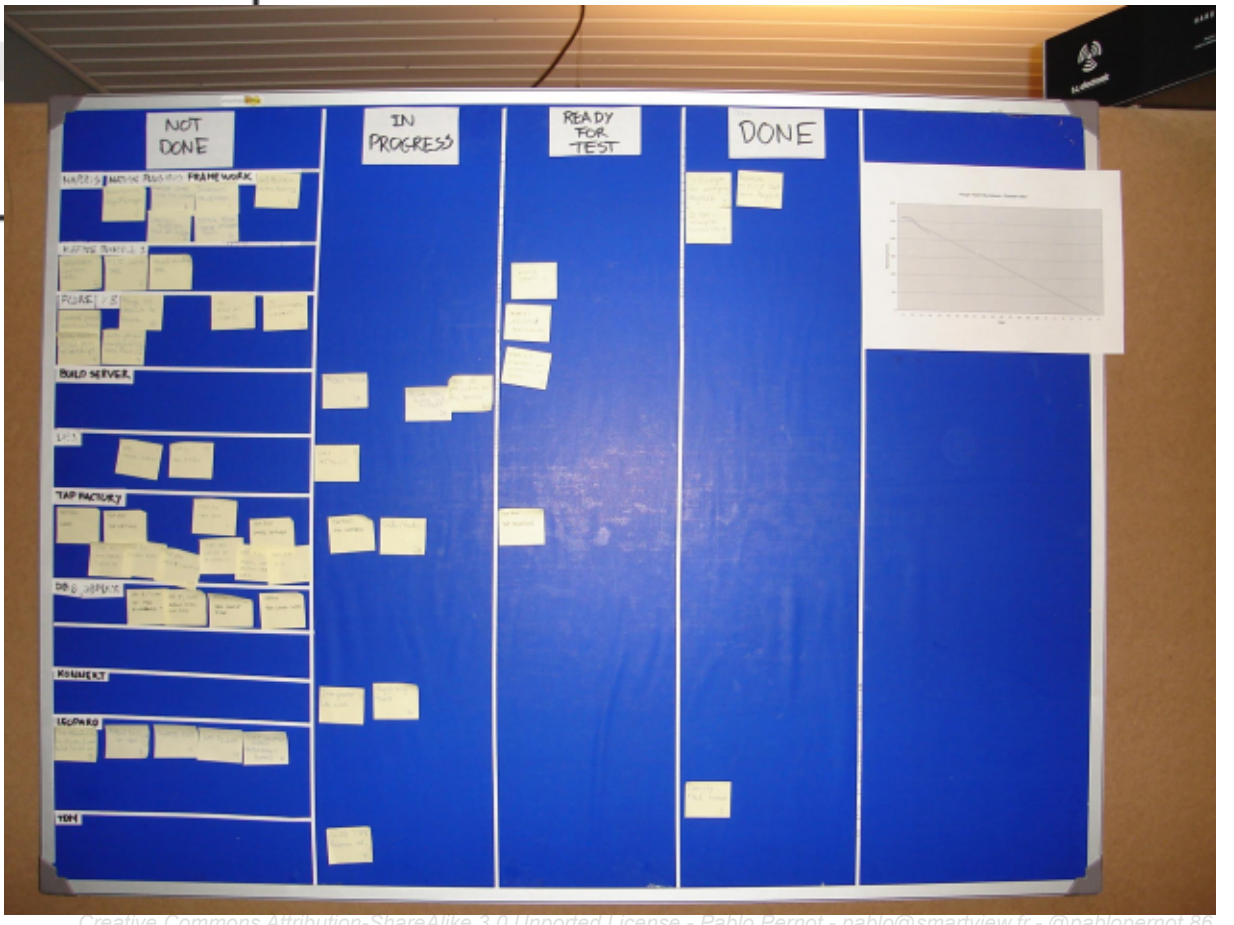
Radiateurs d'information



Source : SCRUM & XP FROM THE TRENCHES
http://www.crisp.se/henrik.../ScrumAndXpFromTheTrenches.pdf

RADIATEURS D'INFORMATION





Burndown Charts

Les burndown charts montrent l'avancement du projet

Il existe 2 types de burndown chart : celui de la release, celui du sprint

Celui de la release se base sur les points de story

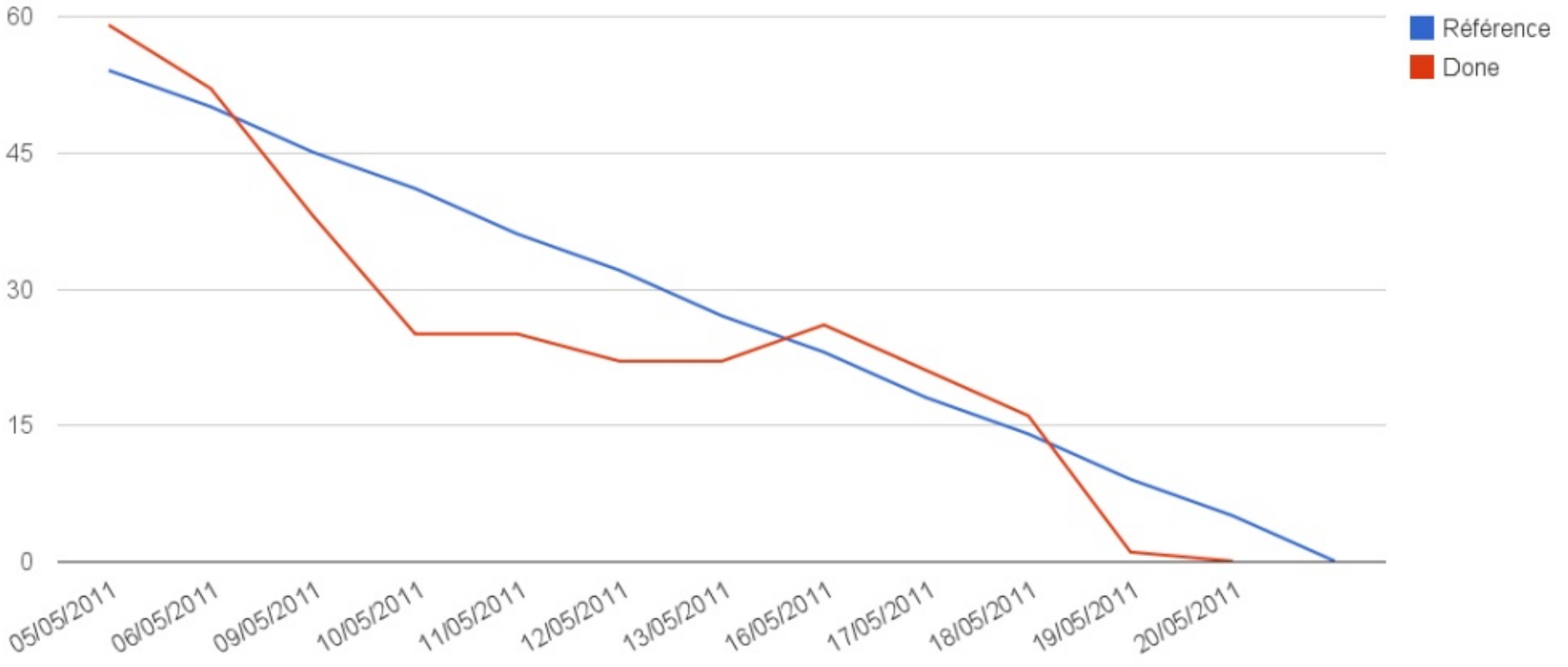
Celui du sprint peut se baser :

- Sur les heures
- Sur les points affectés aux tâches
- Sur le nb de tâches restantes

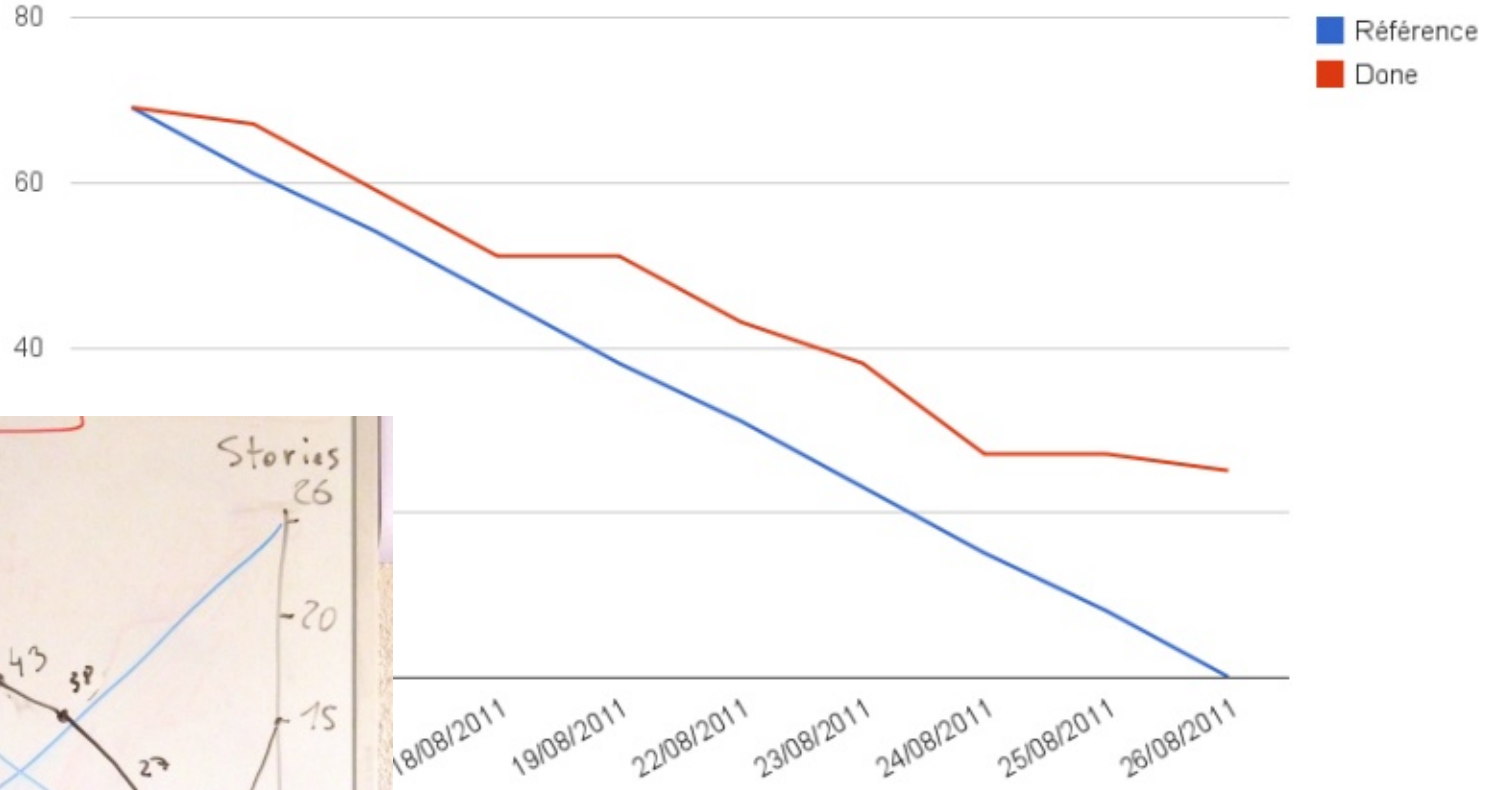
L'avancée du projet est collective

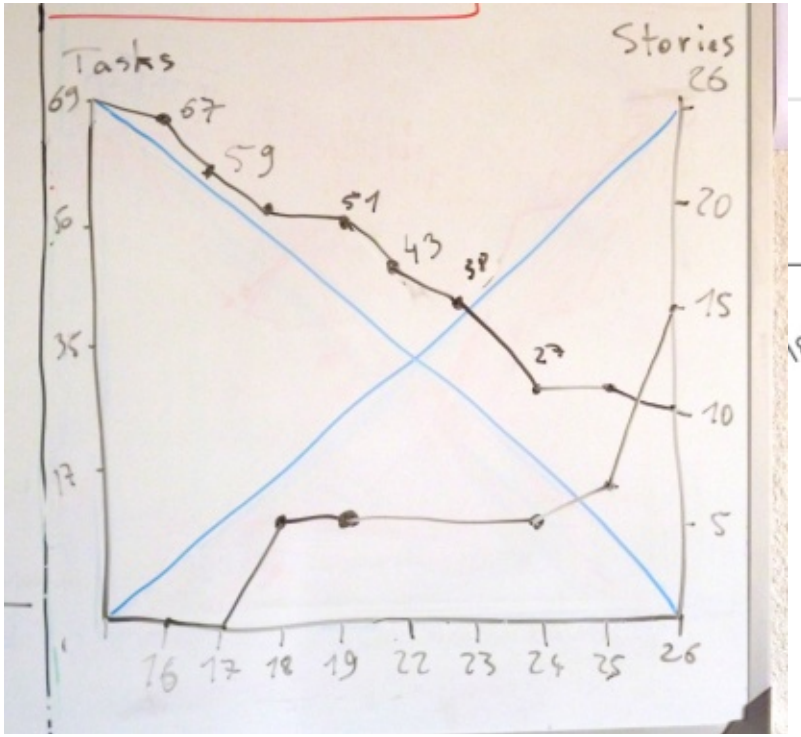
Les burndown charts doivent être visibles par tous (autant par l'équipe que par les parties prenantes).

Burndown Charts



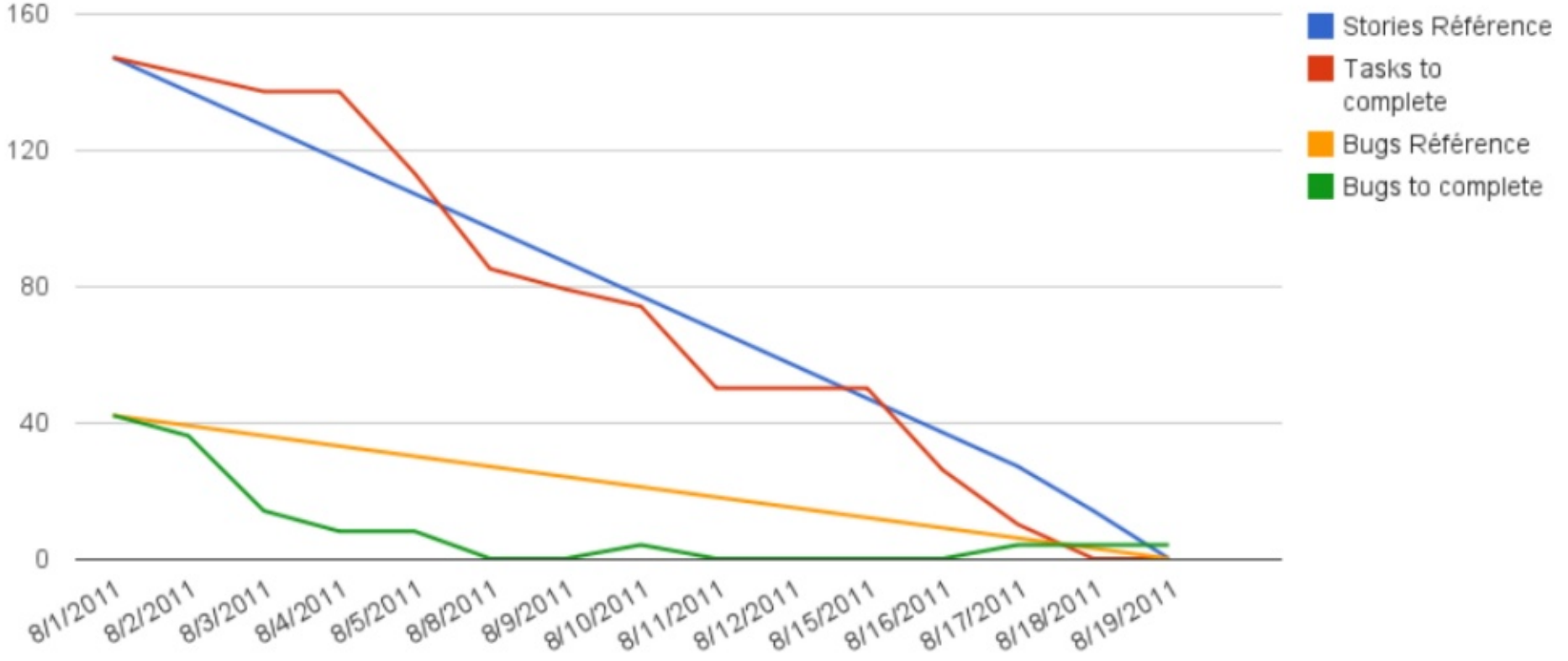
Burndown & burnup charts





Un burndown de points tâches et un burnup de points "story" qui se croisent.
Une bonne pratique.

Burndown charts



Pratique quotidienne : le "daily scrum"

Il dure ~ 15 mn, il a lieu tous les jours, tout le monde est débout (standup meeting), chacun y répond à 3 questions

Qu'est ce que tu as réalisé hier ?

Quelles sont tes tâches aujourd'hui ?

Est-ce que quelque chose te gène dans la réalisation de tes tâches ?

Pratique quotidienne : le "daily scrum"

Ou "Daily Standup"

Il donne de la visibilité à toute l'équipe sur l'ensemble des travaux à intervalles réguliers

Il devrait se dérouler systématiquement à la même heure et dans le même lieu

C'est l'outil de l'équipe, le ScrumMaster, le Product Owner sont optionnels. Le ScrumMaster doit cependant s'assurer que le Daily Scrum se déroule comme convenu mais il doit se faire discret lors du StandUp. Le Product Owner peut profiter de ce moment pour suivre l'avancement du projet. Les parties prenantes peuvent assister mais ne doivent pas intervenir. Seuls les membres de l'équipe peuvent parler.

Il faut trouver un lieu tranquille mais proche des « radiateurs d'informations »

Le Daily Scrum se déroule impérativement debout.

Pratique quotidienne : le "daily scrum"

Il améliore la communication et permet d'éviter d'autres réunions (mais son sujet ne doit pas dévier).

Chacun doit s'exprimer brièvement car le Daily Scrum est « Timeboxé » à 15mn.

Attention de ne pas dévier sur des conversations annexes (règlement d'un point technique complexe, modification des priorités, etc.), ni de régler lors du point les problèmes, il s'agit de les repérer pour les régler ultérieurement.

Chacun s'engage, car l'équipe est responsable. On ne fait pas de compte rendu au Product Owner ou au ScrumMaster (d'ailleurs ces deux rôles ne sont peut-être pas présents)

Ce n'est pas non plus le lieu des règlements de compte (l'équipe doit comprendre qu'elle doit s'unir pour réussir)

Par moment quand un Daily Scrum dérape le ScrumMaster peut se tourner face au mur et appuyant ses mains contre celui-ci (« spread eagle ») pour indiquer que ce qui se déroule est inutile et n'est pas le but du Daily Scrum.

Atelier : "Scrum from Hell"

atelier
"Scrum from Hell"

Source : http://xp123.com/g4p/0410b/index.htm

Revue de sprint

On présente le résultat du Sprint (toutes les User Story achevées)

C'est le moment durant lequel le Product Owner et les parties prenantes (stakeholders) inspectent le produit en détails en fonction de l'objectif du Sprint et la vision du produit et prennent les décisions d'adaptation nécessaires à la réussite de l'objectif du projet.

Il peut y avoir de nombreux invités à cette occasion

Il faut préparer la réunion (scénariser) et rappeler les objectifs du Sprint en premier lieu

A la fin de la démo on peut calculer la vélocité effective du Sprint et donc réajuster le plan de release

A la fin de la démo le Product Backlog est actualisé (User Story achevées, nouvelles User Story émergentes, anomalies, etc.)

Les BurndownChart sont réinitialisés ou remis à jour

RÉTROSPECTIVE

Elle est là pour améliorer les processus
(identifier les axes d'amélioration, capitaliser sur les bonnes pratiques), faire un état des lieux du sprint qui vient de s'achever

Comment améliorer les choses ?
Qu'est ce qui a bien fonctionné ?
Qu'est ce qui a mal fonctionné ?
Comment résoudre les problèmes que nous voyons apparaître ?

Il faut préparer une rétrospective (la préparation peut durer aussi longtemps que la rétrospective elle-même)

C'est le ScrumMaster qui va faciliter et organiser cette réunion

C'est le ScrumMaster qui devrait savoir si une amélioration est réellement nécessaire ou pas, il ne va pas choisir mais il sera un arbitre déterminant.

A la fin de la rétrospective on a un plan d'action

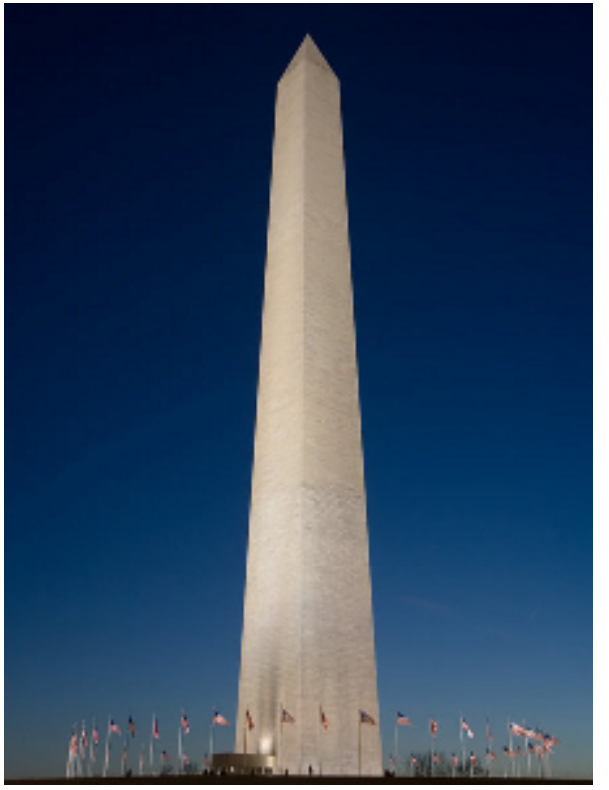
Atelier : "Retrospective : speed boat"

atelier
Rétrospective & speed boat

Les 5 pourquoi

L'obstacle n'est pas toujours celui que l'on croit : Concept des 5 pourquoi

Le « Washington Monument » s'érode et la firme responsable du ciment ne réussit pas à en trouver la cause (« root cause analysis »).



Pourquoi le bâtiment se désagrège-t-il ?
Parce que l'on y applique trop de produits chimiques

Pourquoi applique-t-on trop de produits chimiques ?
Pour nettoyer les crottes de pigeons !

Pourquoi y a-t-il autant de pigeons ?
Car ils mangent les insectes sur le bâtiment !

Pourquoi y-a-t-il autant d'insectes ?
A cause de la lumière !

Solution : Réduire les horaires d'éclairages du Monument...

Quelques questions

Faites 2 groupes et travaillez chacun sur ces deux sujets avec l'aide des "5 pourquoi", observons les résultats

Il y a souvent trop de recettes à réaliser à la fin de chaque sprint !

Le product owner n'est pas assez présent avec l'équipe !

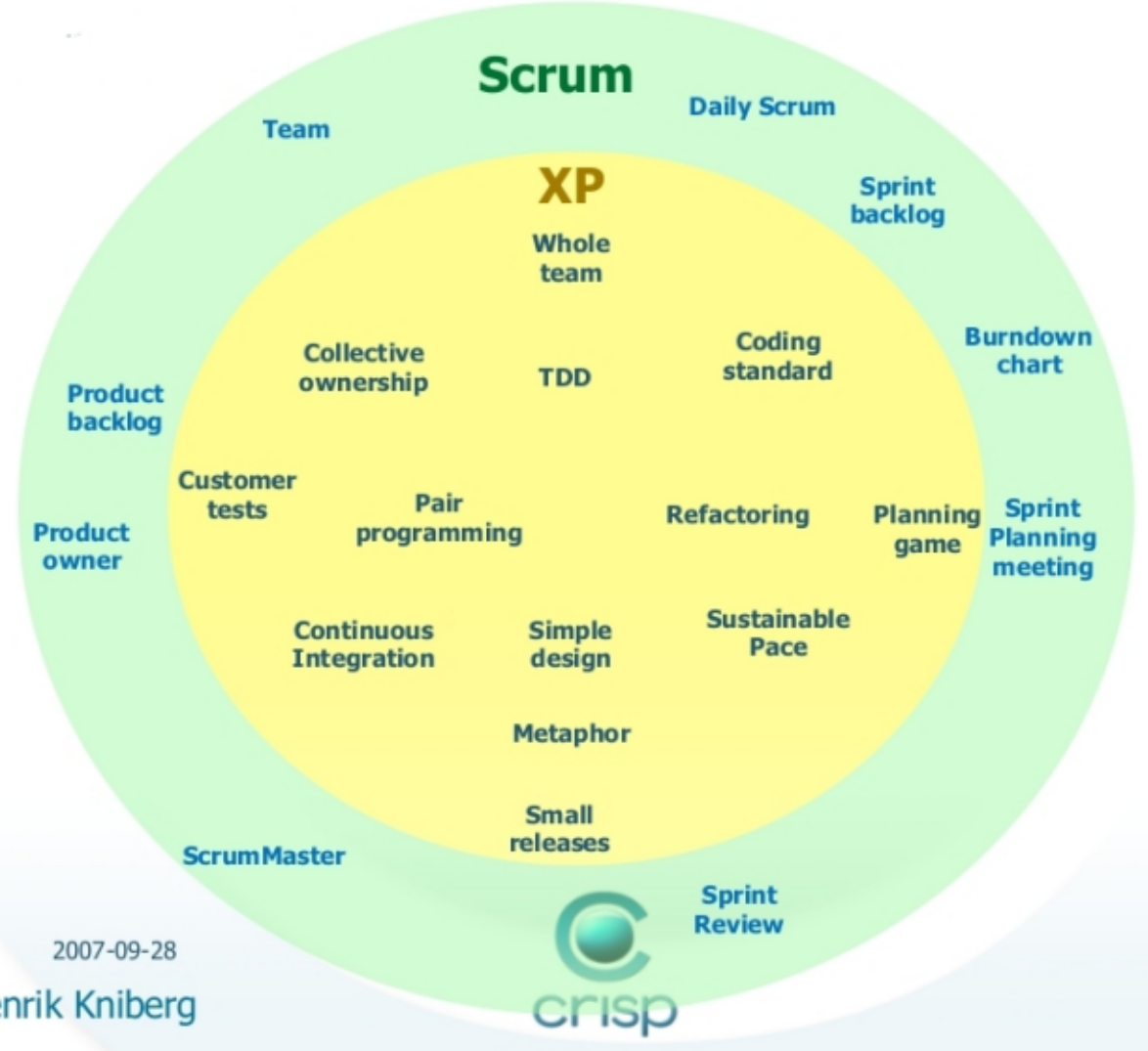
Le daily-stand up n'est jamais achevé !

Il n'y a pas assez de développeurs/testeurs dans l'équipe !

On dépend trop de développeurs externes à l'équipe !

On travaille sur trop de projets !

Pratiques d'ingenierie : eXtreme Programming

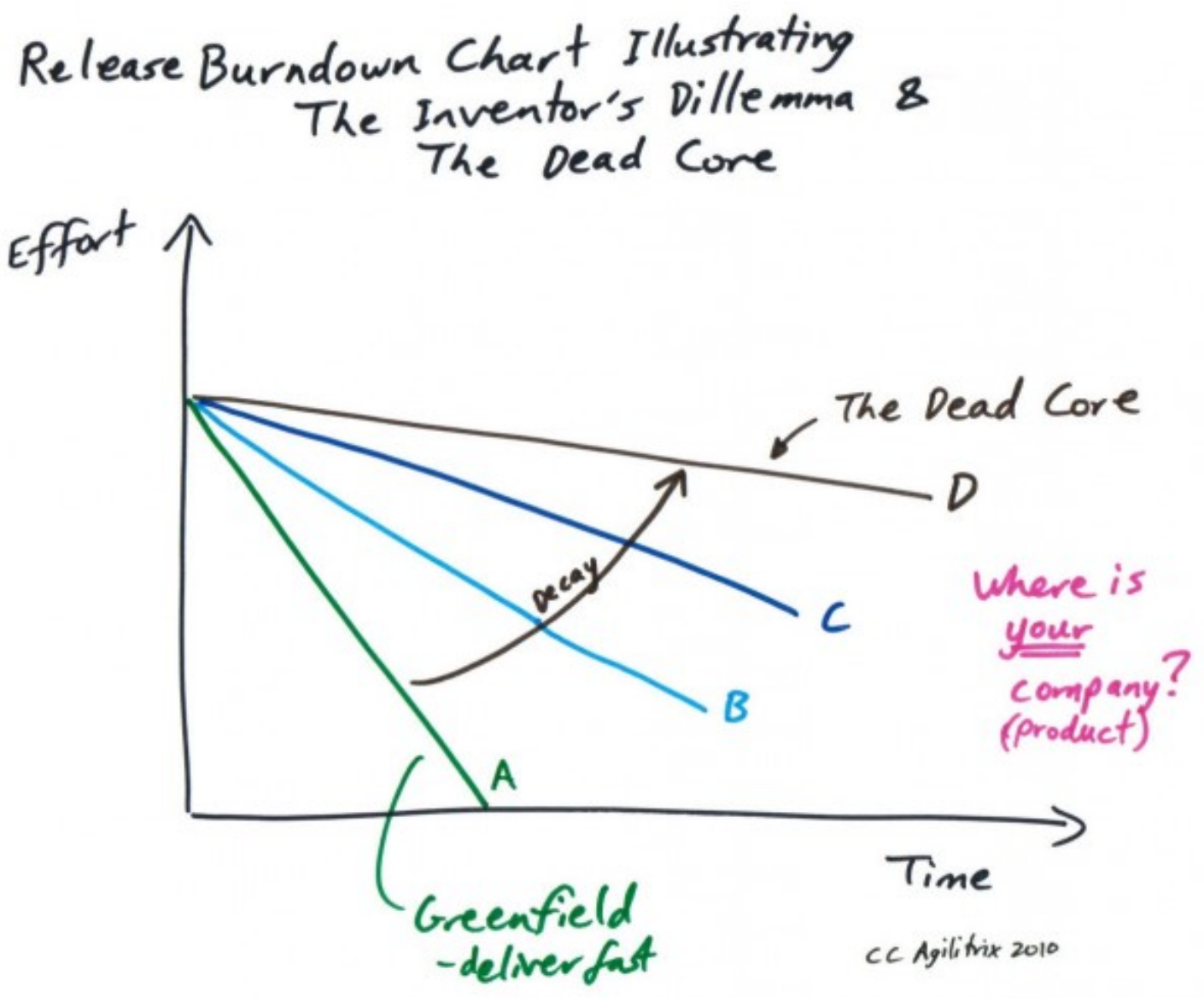


Scrum et XP devraient dans la pluspart des projets informatiques être indissociables

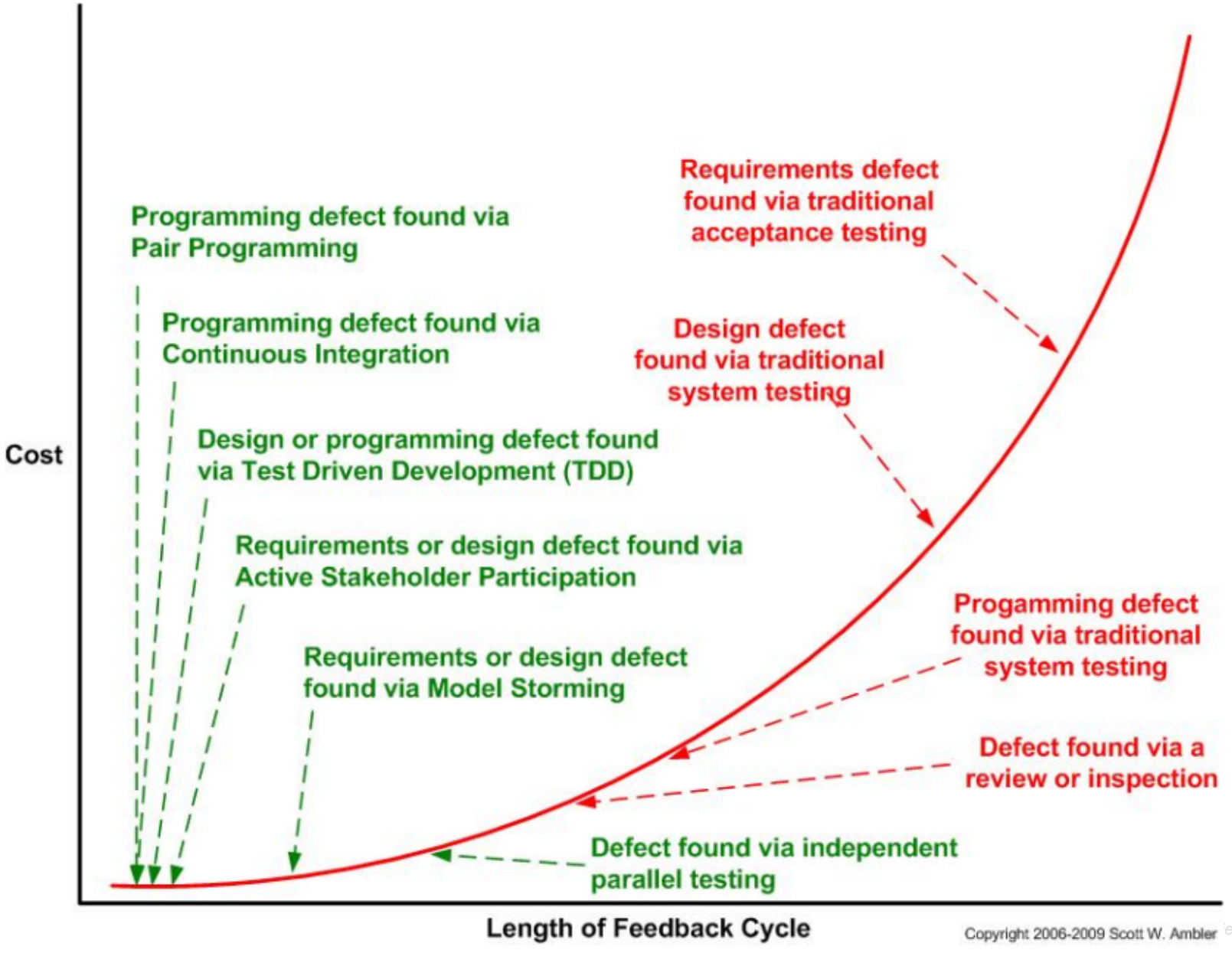
Dette technique

Astérix chez les hélvètes, Uderzo & Goscinny

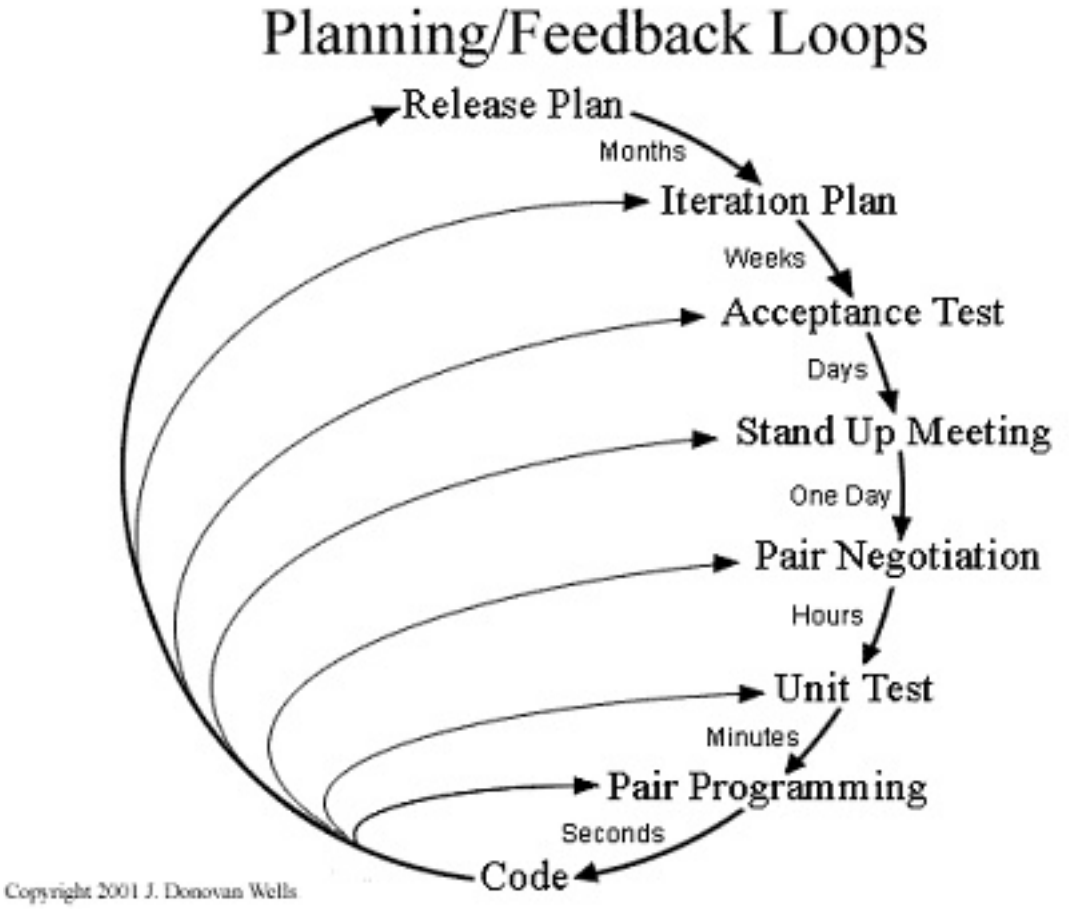
DETTE TECHNIQUE



Feedback : Coût du changement



FEEDBACK, FEEDBACK



SIMPLICITÉ

La complexité ne doit pas s'imposer

Le Mythe de l'ultra-générique
Optimisations de performance inutiles
Etc.

C'est la simplicité qui doit s'imposer…

pas la facilité !
Différence : la qualité n'est pas négociable !

Mais

On favorise l'émergence

Simplicité

YAGNI

You ain't gonna need it

KISS

Keep it simple stupid

DRY

Don’t repeat yourself

Fail Fast

Courage & respect

Quel niveau de confiance j'accorde à mes partenaires ?

De 1 à 5

1 : je n'ai aucune confiance

5 : j'ai une confiance aveugle

Comment je juge le niveau de confiance que m'accordent mes partenaires ?

A quel point suis-je à l'aise pour parler à mes partenaires des problèmes et points de blocage que je rencontre ?

Vous êtes Scrummaster l'un des membres de l'équipe vient voir pour expliquer qu'il ne supporte plus une personne de l'équipe pour telle ou telle raison, quel serait votre positionnement ?

Focus sur des pratiques : Pair programming

Idées reçues
Les managers voient cela comme une perte de temps

Les développeurs ont été éduqués pour travailler de façon isolée

Certains Senior pensent que travailler ainsi va les ralentir, ou que cela va compliquer les choses

Mais il apparaît que :

Cela facilite le transfert de connaissance (sans que chacun perde sa spécialité)

La qualité du code est drastiquement améliorée (c'est la meilleure revue possible) : beaucoup d'anomalies pernicieuses chronophages détectées ensuite

les débutants sont incorporés sans problème

La productivité est a minimum égale, souvent meilleure

Les gens ont plaisir à faire du pair programming

Focus sur des pratiques : Tests automatisés

Un développement est moderne et mature si il automatise ses tests

Permet d'avoir des garanties sur la qualité du code

Permet aux développeurs de se partager le code sans risque, sans crainte

Les tests peuvent servir de documentation

Erèirra ne recnava

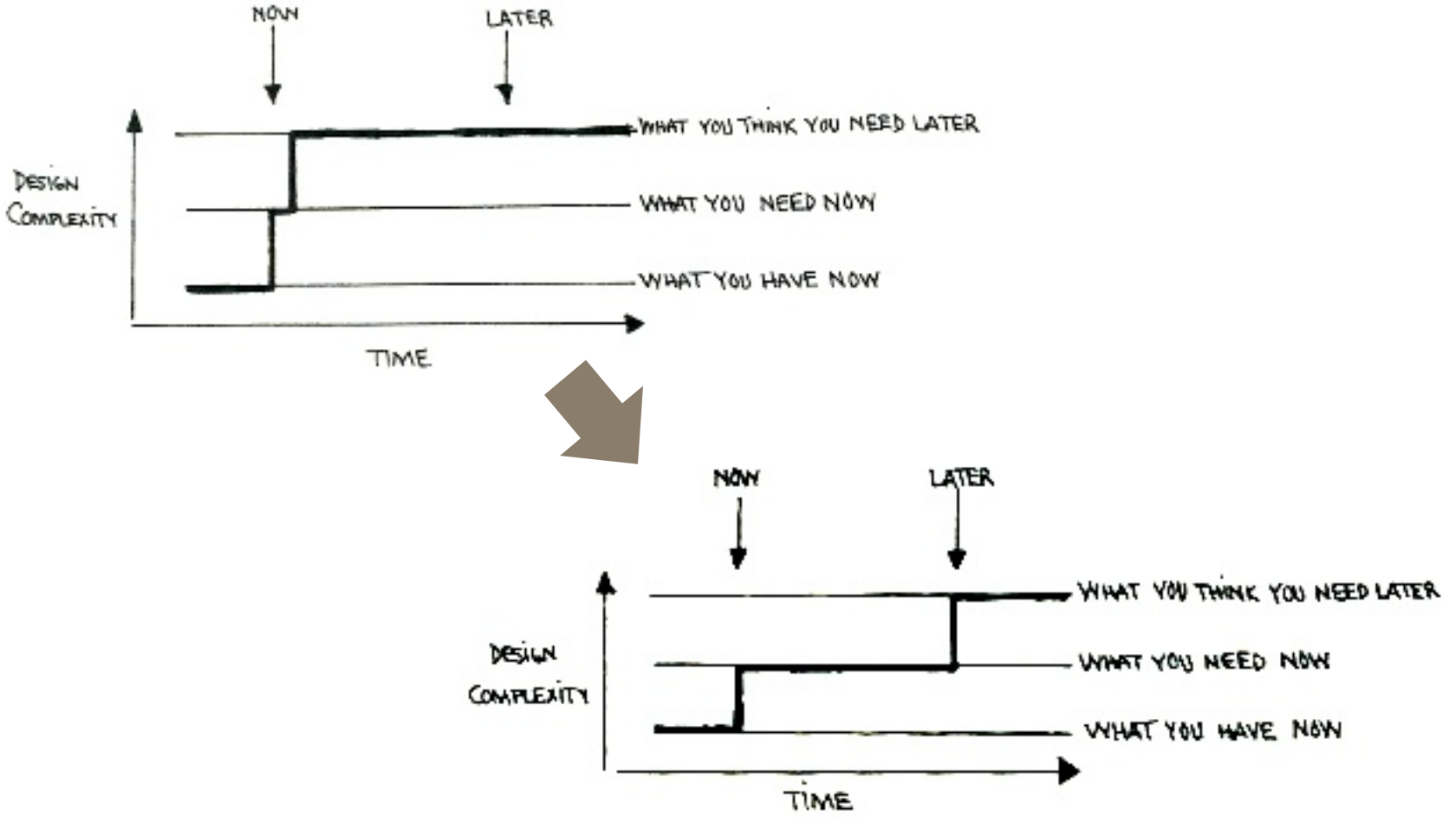
Test Driven Development

Définir l'object
Ecrire le test et le faire échouer
Ecrire le code
Le test fonctionne

Un exemple de Test Driven Development proposé par pytddmon
http://www.youtube.com/watch?v=sD6qzJNQEpE

Le TDD propose une alliance entre
les deux éléments pour lesquels on observe le plus de réticence sur le terrain de la part des développeurs
- conceptualiser et découper son développement
- faire des tests

Focus sur des pratiques : Refactoring



Extreme Programming Explained, embrace change, Kent Beck

Focus sur des pratiques : Refactoring

Refactoriser c'est changer un logiciel de façon à ce que comportement ne varie pas mais que sa structure interne soit meilleure

On ne décide pas de refactoriser, c'est un travail sous-jacent, constant.

Penser le code comme devant/pouvant être revisé

Ne pas développer des choses inutiles

Ne pas développer des choses trop complexes

L'un des objectifs du refactoring est de rendre le code plus lisible, plus compréhensible, plus abordable pour le développeur suivant

Focus sur des pratiques : Refactoring

Modularisez !

```
void printOwing(double amount)
{
      printBanner();

      //print details
       WriteLine("name:" + _name);
       WriteLine("amount" + amount);
  }
```

http://www.refactoring.com/catalog/index.html

Inline Method,
Replace method with method object
Hide delegate

etc..

```
void printOwing(double amount)
  {
     printBanner();
     printDetails(amount);
  }
  void printDetails(double amount)
  {
      WriteLine("name:" + _name);
      WriteLine("amount" + amount);
  }
```

"Refactoring Tips by Martin Fowlers", Igor Crvenov

Focus sur des pratiques : Refactoring

Quand refactoriser ?

Quand vous ajoutez une fonction

Quand vous corrigez une anomalie

Quand vous faîtes une revue de code

... Et avant de refactoriser il faut une solide suite de tests pour ne pas introduire d'anomalies

Quand ne pas refactoriser ?

Quand le code est dans un trop mauvais état et qu'il faut tout reprendre

Quand une deadline approche et que l'effort de refactorisation n'a plus assez de valeur

"Refactoring Tips by Martin Fowlers", Igor Crvenov

Focus sur des pratiques : intégration continue

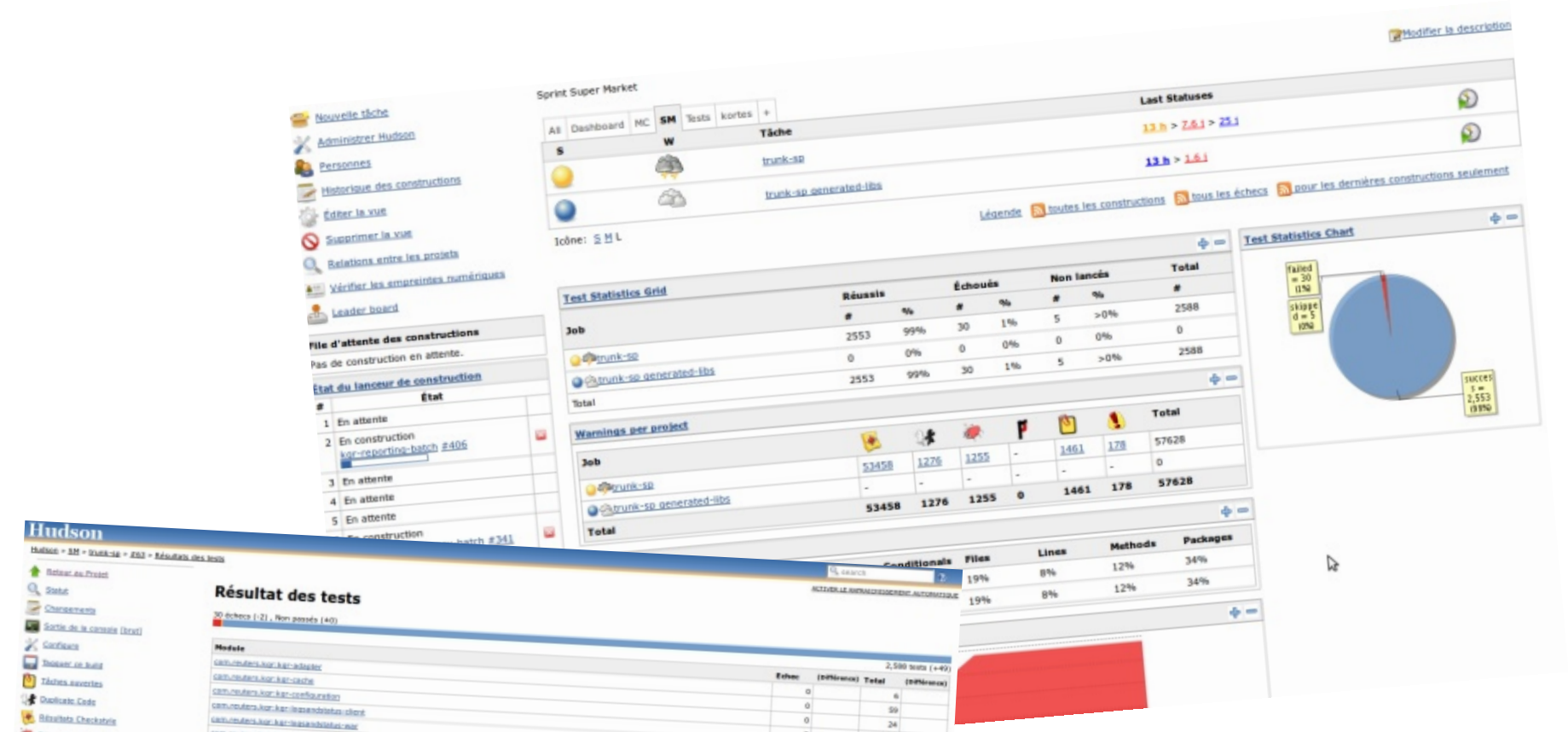
Quelques règles :

n°1 : la première des priorités est de réparer le build cassé
Pas de fatalité ! Rigueur !

Exécution toutes les nuits !
- A chaque commit de code !!
- Attention à la lenteur du build !!!

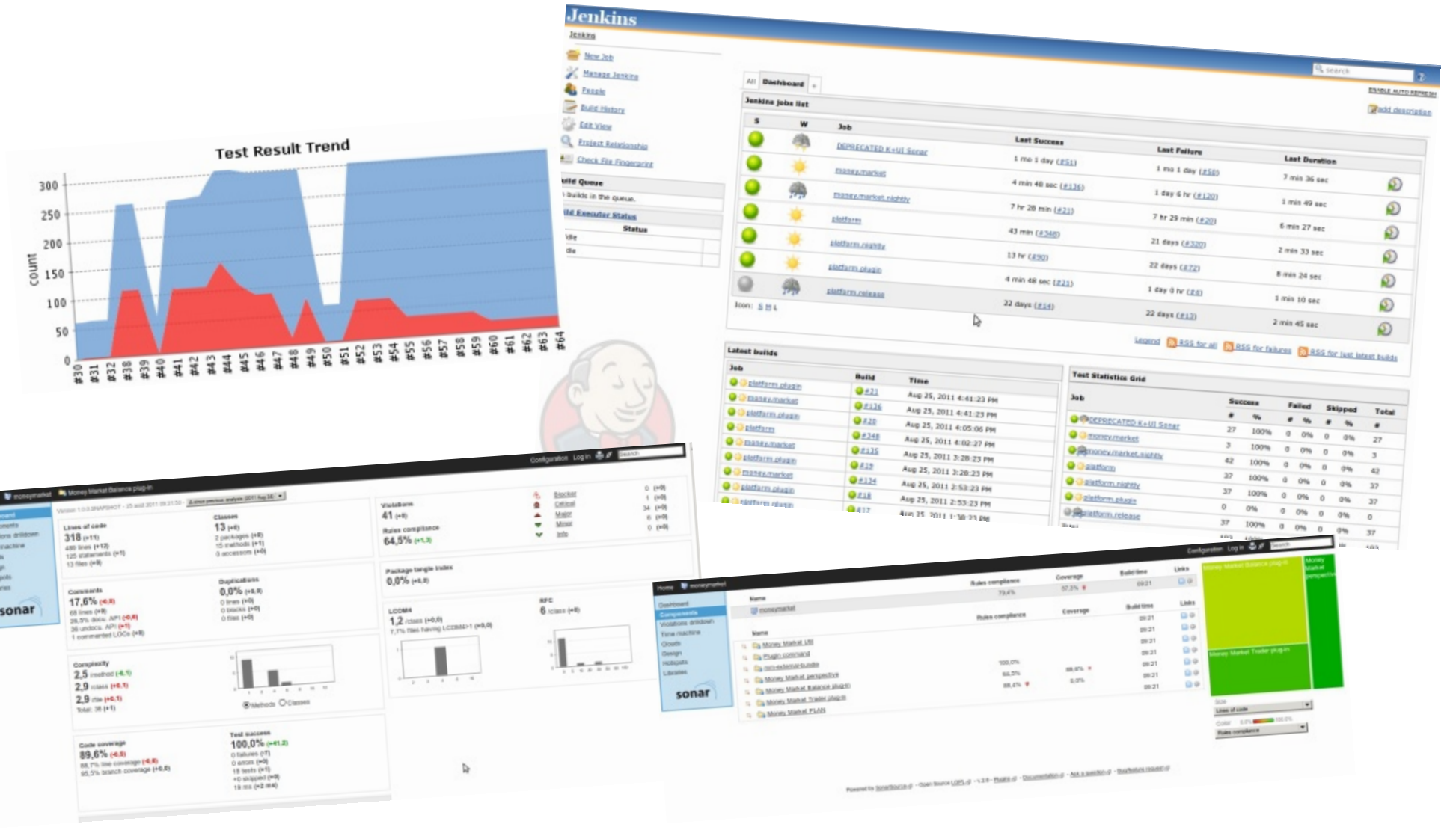
Gestion des versions avec des équipes multiples
http://www.infoq.com/articles/agile-version-control

Focus sur des pratiques : intégration continue



Il est nécessaire de déployer une plate-forme d'intégration continue de type Jenkins/Hudson, Sonar, TFS, etc.

Focus sur des pratiques : intégration continue



Atelier "XP Game"

atelier
"XP Game"

Une marque de maturité ? KanBan

David Anderson : "Scrum ne règle pas tous les problèmes, ni ne s'applique dans toutes les situations" (est-ce vrai ?)

Peut-on faire du scrum avec la "Care & Maintenance" ? Le support ? Le HelpDesk ? Gestion des services IT ? Lié à "DevOps" ?

Pour embrasser l'intégralité de ce qui est réalisé ?

KanBan

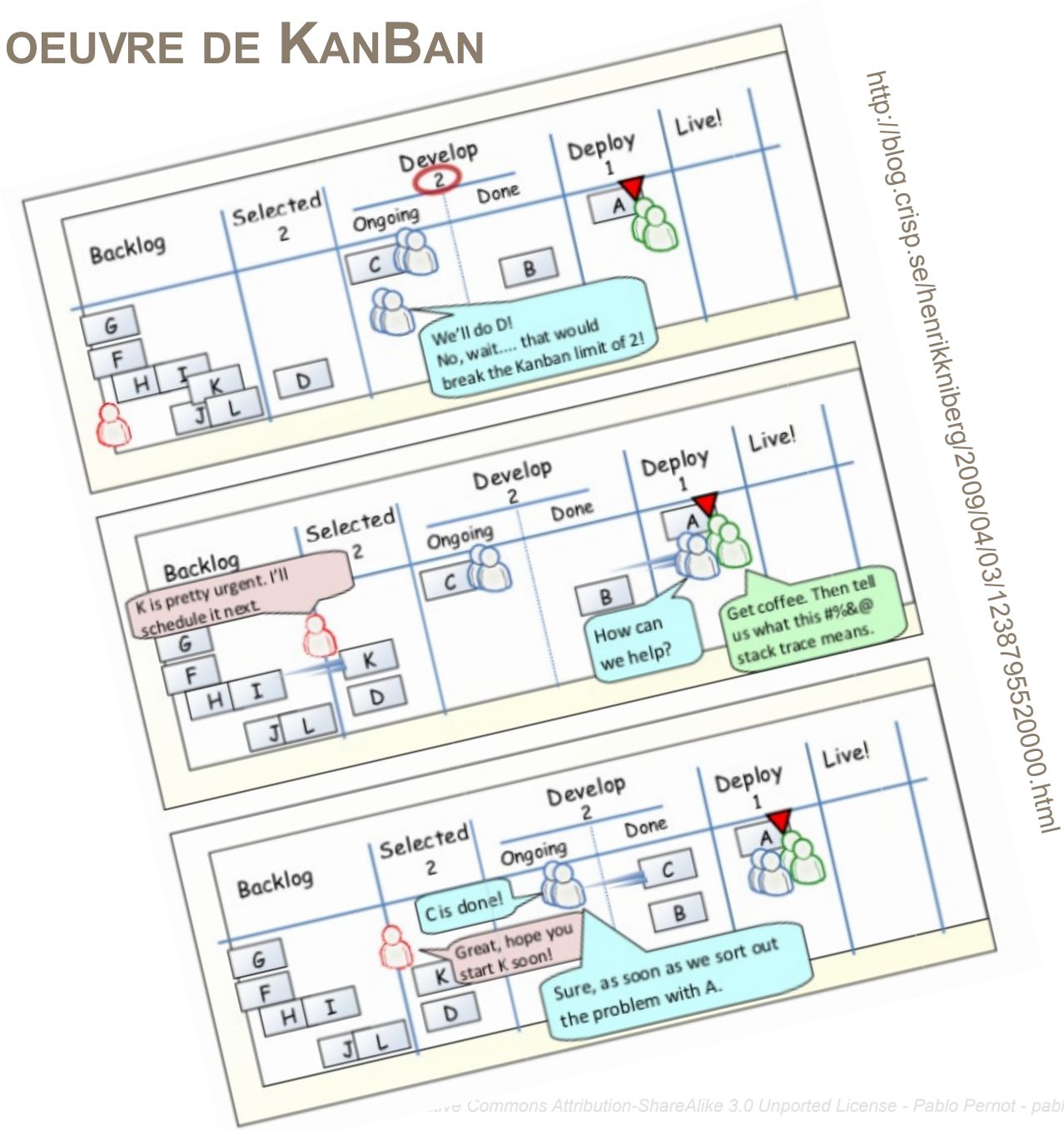
Visualiser le flux

Limiter le travail en cours (Work in Progress)

Gérer le flux

Focus sur la qualité

Mise en oeuvre de KanBan



http://blog.crisp.se/henrikkniberg/2009/04/03/1238795520000.html

Mise en oeuvre de KanBan

Apprenez à connaitre votre système

Identifiez et priorisez vos sources

Définissez votre processus

Concevez votre tableau de flux

Définissez les limites

Décidez des rôles

Décidez des réunions

http://www.fabrice-aimetti.fr/dotclear/public/traductions/demarrer-en-kanban.pdf

Réunions pour Kanban

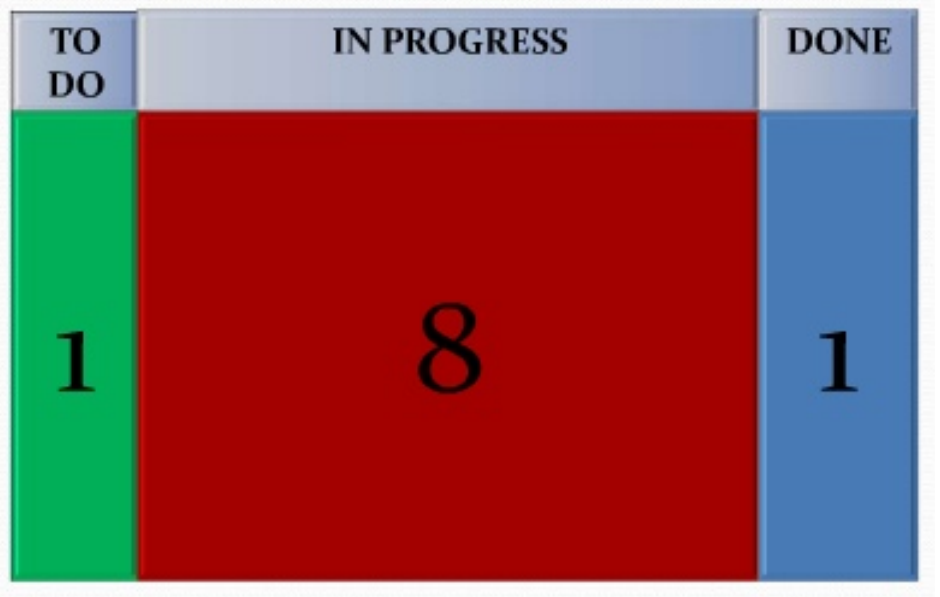
Réunion de définition, d'expression du besoin

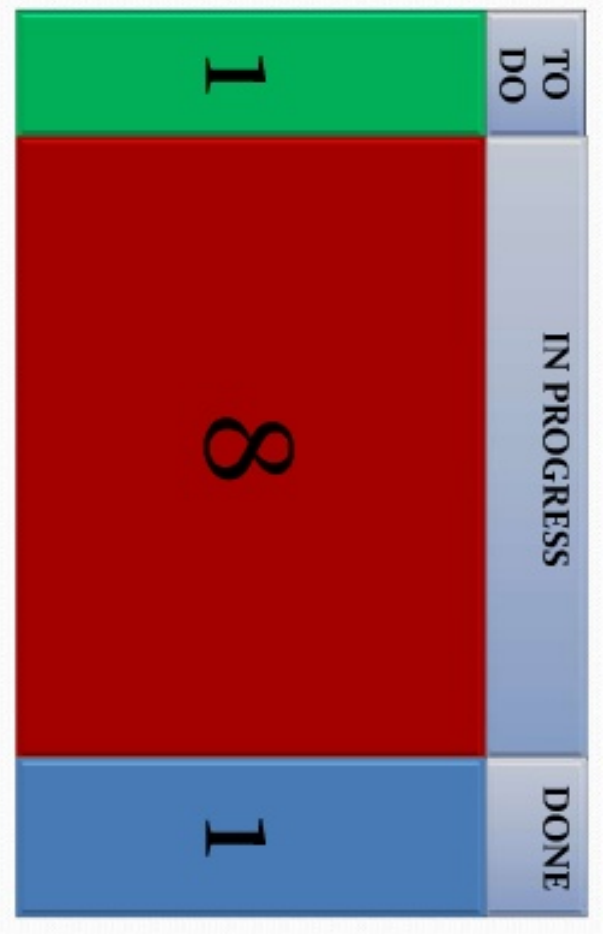
Réunion quotidienne

Réunion d'apprentissage, d'amélioration continue et d'analyse du feedback

http://www.fabrice-aimetti.fr/dotclear/public/traductions/demarrer-en-kanban.pdf

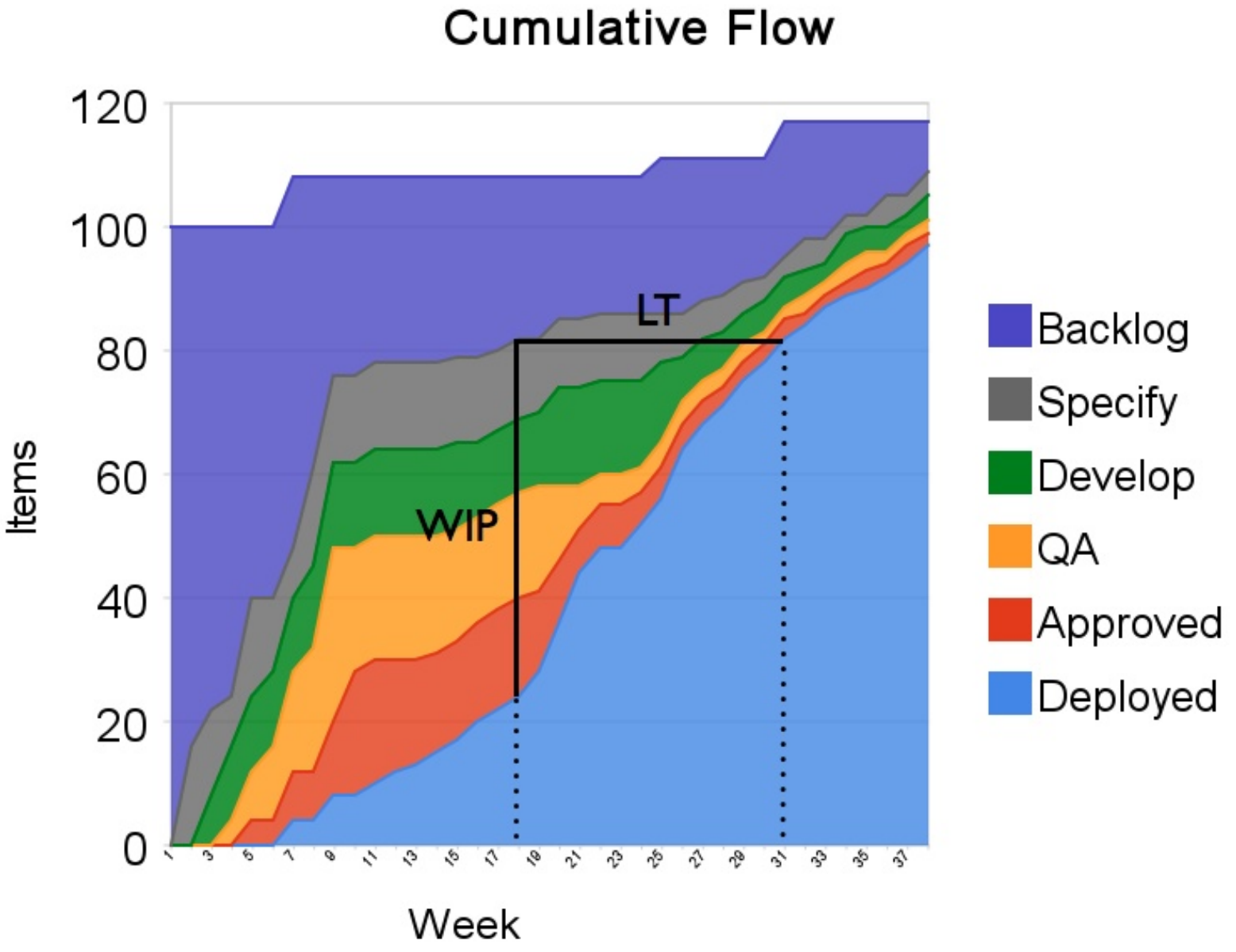
Visualiser le flux



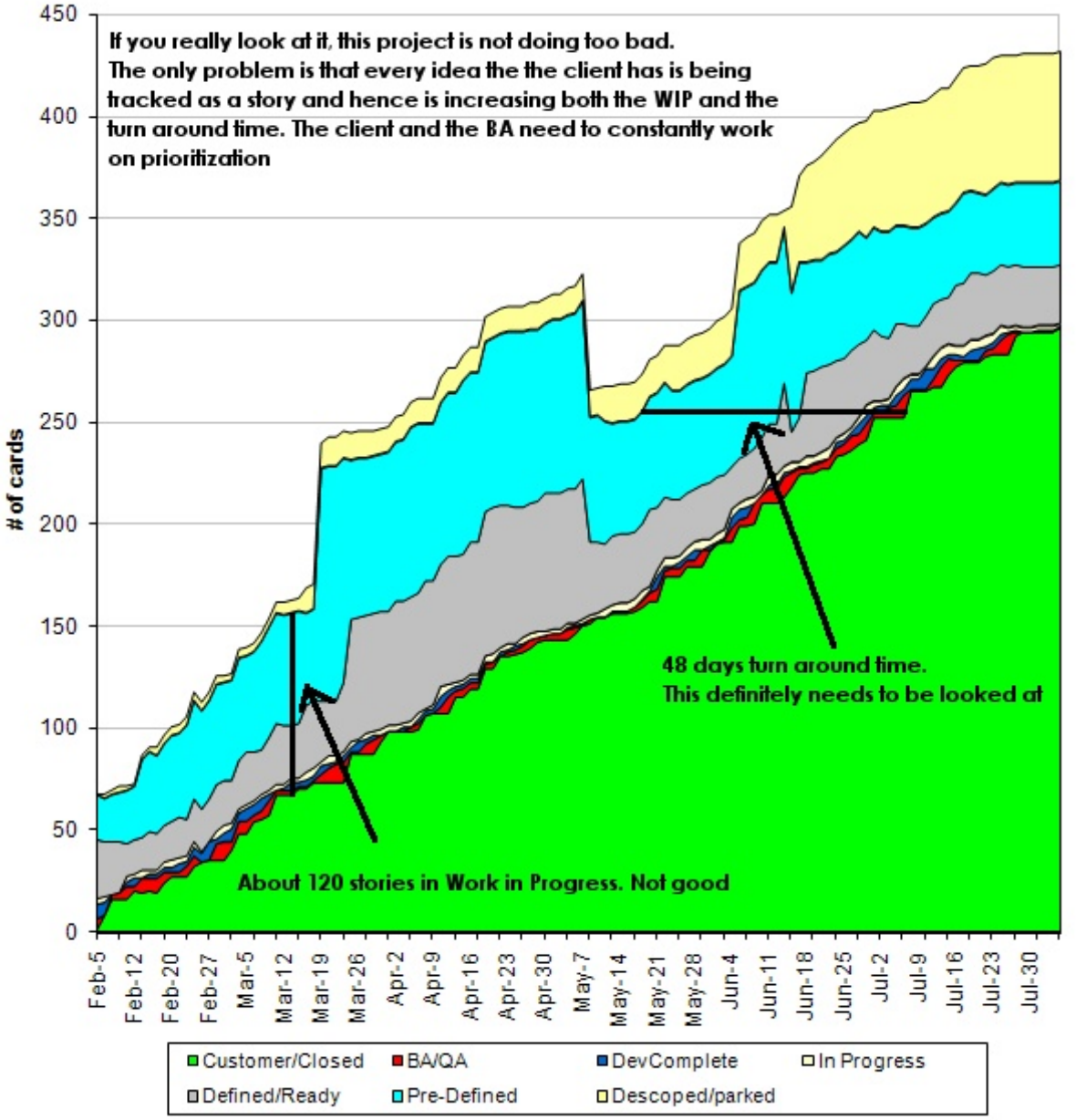


http://www.slideshare.net/yyeret/explaining-cumulative-flow-diagrams-cfd

Visualiser le flux



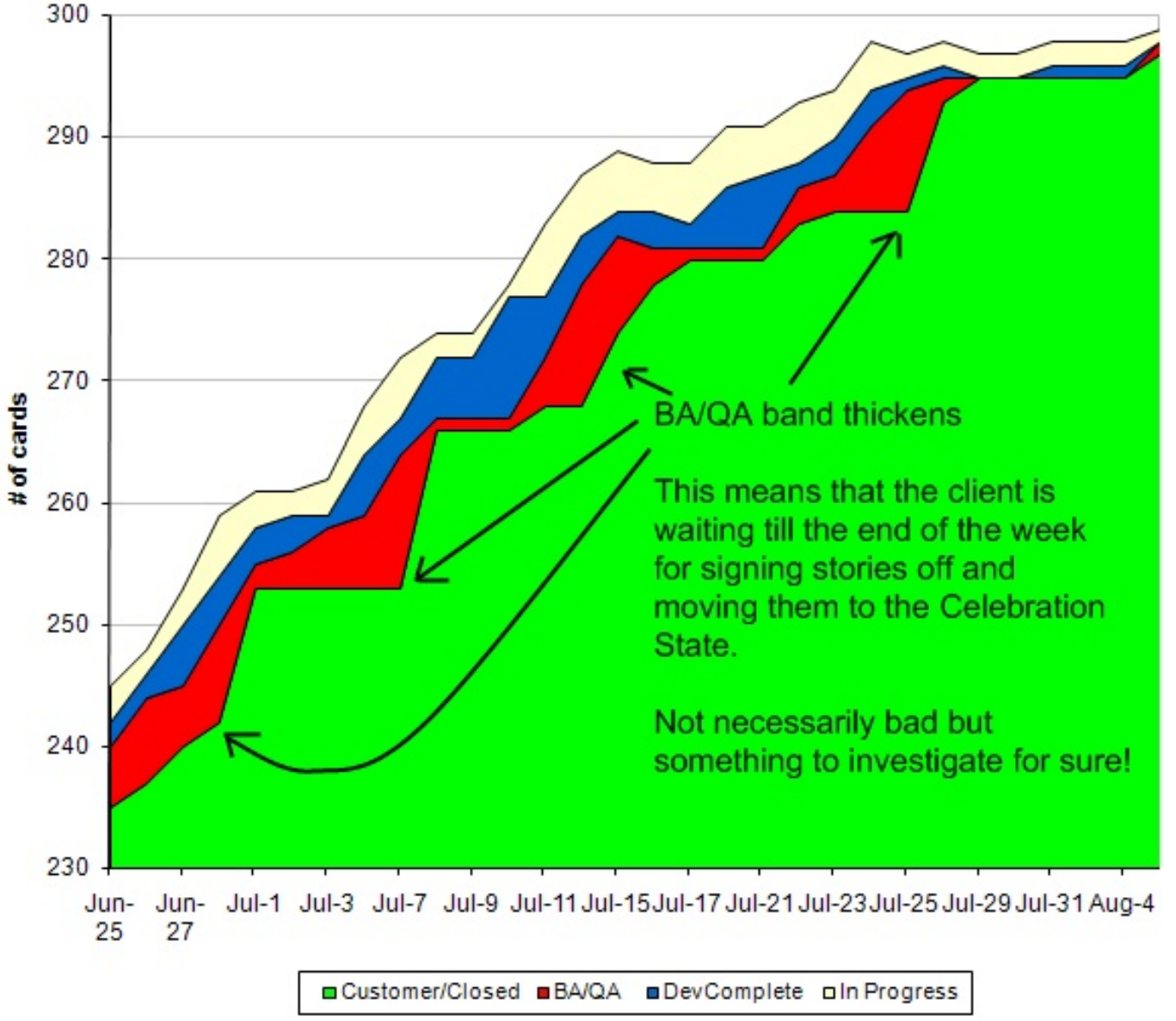
VISUALISER LE FLUX



Trop de travail à faire signifie probablement :

- vous ne finissez pas ce qui est engagé
- une étape ultérieure ne peut pas intégrer votre travail

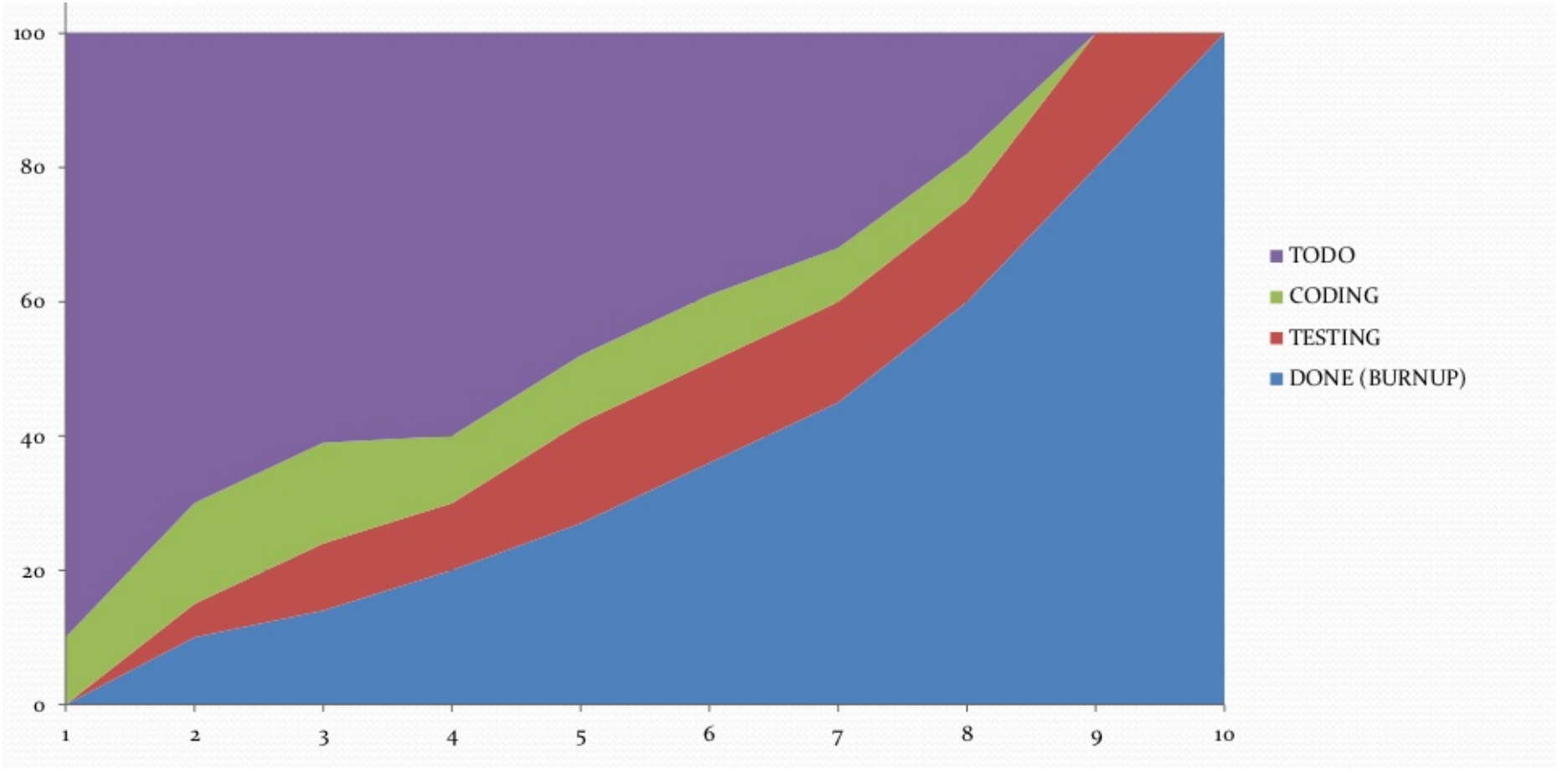
VISUALISER LE FLUX



http://blog.akshaydhavle.com/2008/12/cumulative-flow-diagrams/

Visualiser le flux

Mieux !



http://www.slideshare.net/yyeret/explaining-cumulative-flow-diagrams-cfd

VISUALISER LE FLUX : DÉFINITION DE CLASSE DE SERVICE

http://www.slideshare.net/deimos/david-anderson-kanban-at-q-con

Quelques débats sur KanBan

Points de vigilance

Kanban est-il "people friendly" ?

Kanban permet-il un véritable changement ? N'est-il pas trop statique, trop conforme ?

Kanban est-il Scrum sans itération ?

Faut-il avoir fait du scrum avant de passer à Kanban ?

Est-ce que les estimations sont une perte de temps ?

David Anderson décrit Kanban comme "proposant un changement évolutif plutôt qu'une révolution (comme agile)"

Software-Enginering Radio, Podcast, Francfort, Germany

http://www.slideshare.net/deimos/david-anderson-kanban-at-q-con

Atelier : KanBan Game

atelier
"Kanban Game"

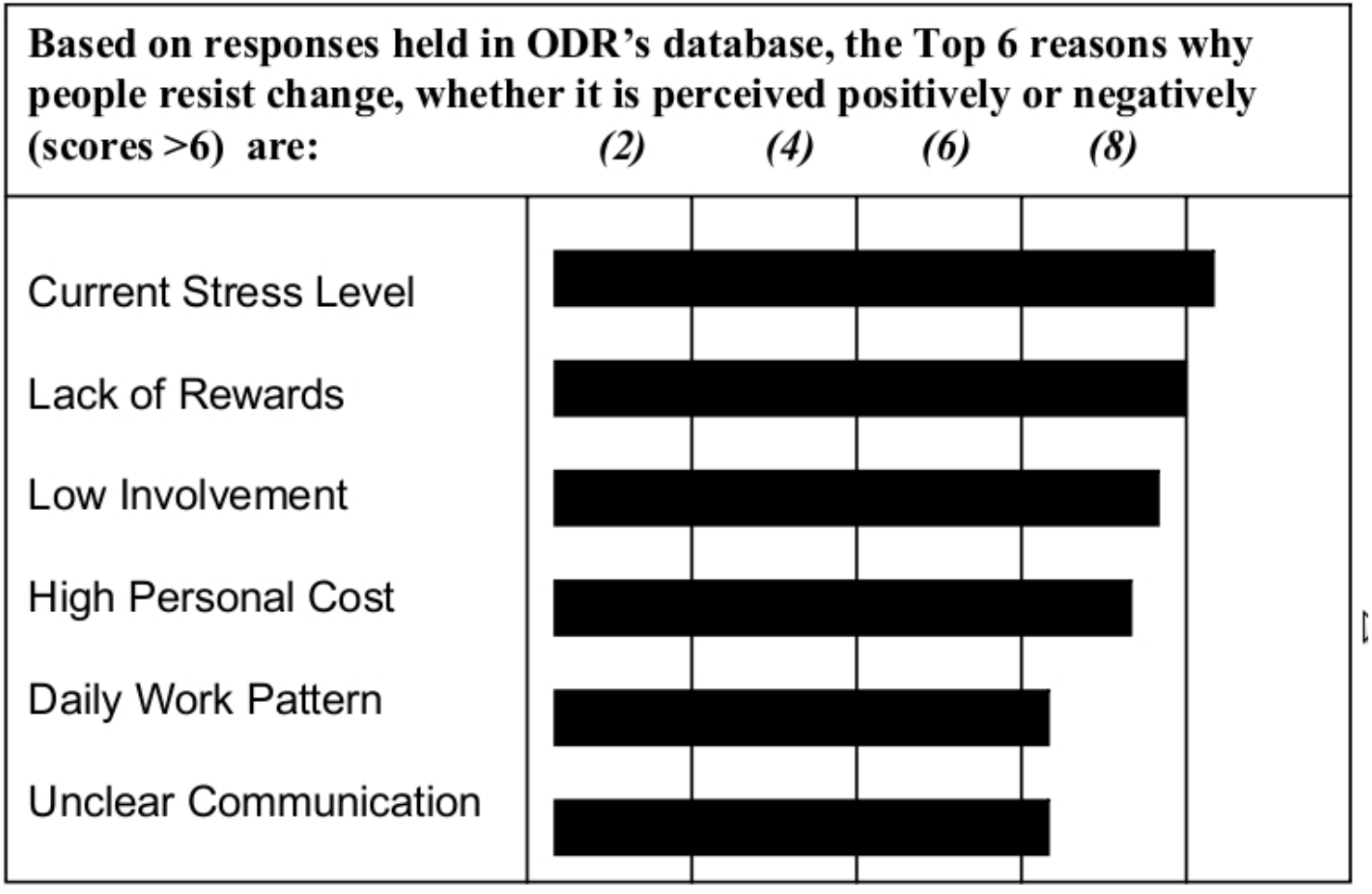
Transition Agile en entreprise : ADAPT

Awareness
Desir
Ability
Promotion
Transfer

L'agile est avant tout un processus de conduite du changement

Management & conduite du changement

En introduisant Scrum dans votre société vous allez bousculer les habitudes, provoquer le changement, attention d'être attentif aux facteurs bloquants



Gérer le changement

L'agile, surtout Scrum, va mettre en évidence les problèmes et les dysfonctionnement.

Il joue le rôle d'un révélateur. Il devient donc aussi une cible idéale.

A chaque fois que vous abandonnez ou détournez une pratique Scrum pour vous conformer à l'entreprise (ou pour faciliter son adoption) vous perdez de la valeur (vous devenez des « ScrumBut »)

> Il faut donc l'éviter autant que possible
>
> C'est toujours le signe d'une défaillance de fonctionnement de l'entreprise et autant s'attaquer à la source

Ken Schwaber estime que seulement 35% des entreprises qui essayent Scrum réussissent. Pourquoi les autres échouent ? Car elles refusent de régler les problèmes que Scrum rend visibles.

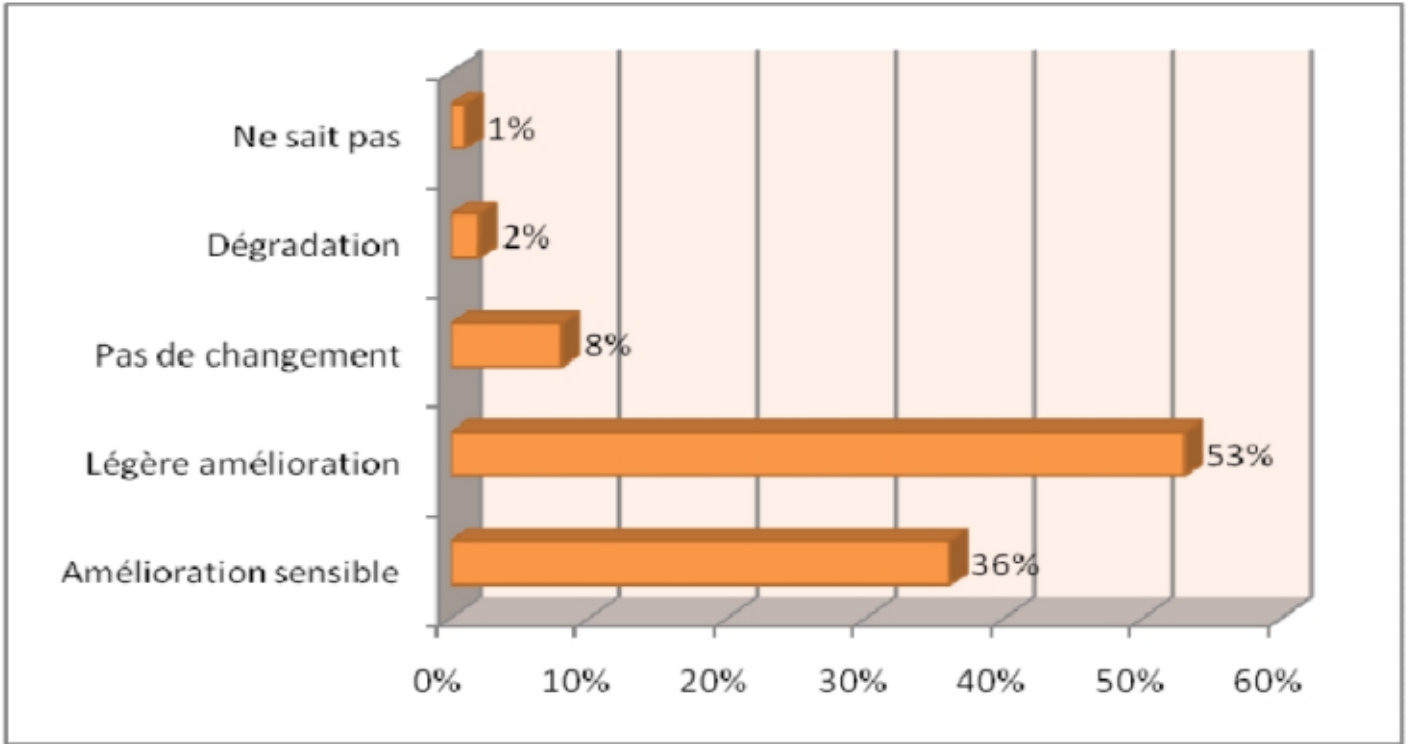
Cependant

> Un bon ScrumMaster est un ScrumMaster vivant, un ScrumMaster mort

Gérer le changement : Motivation

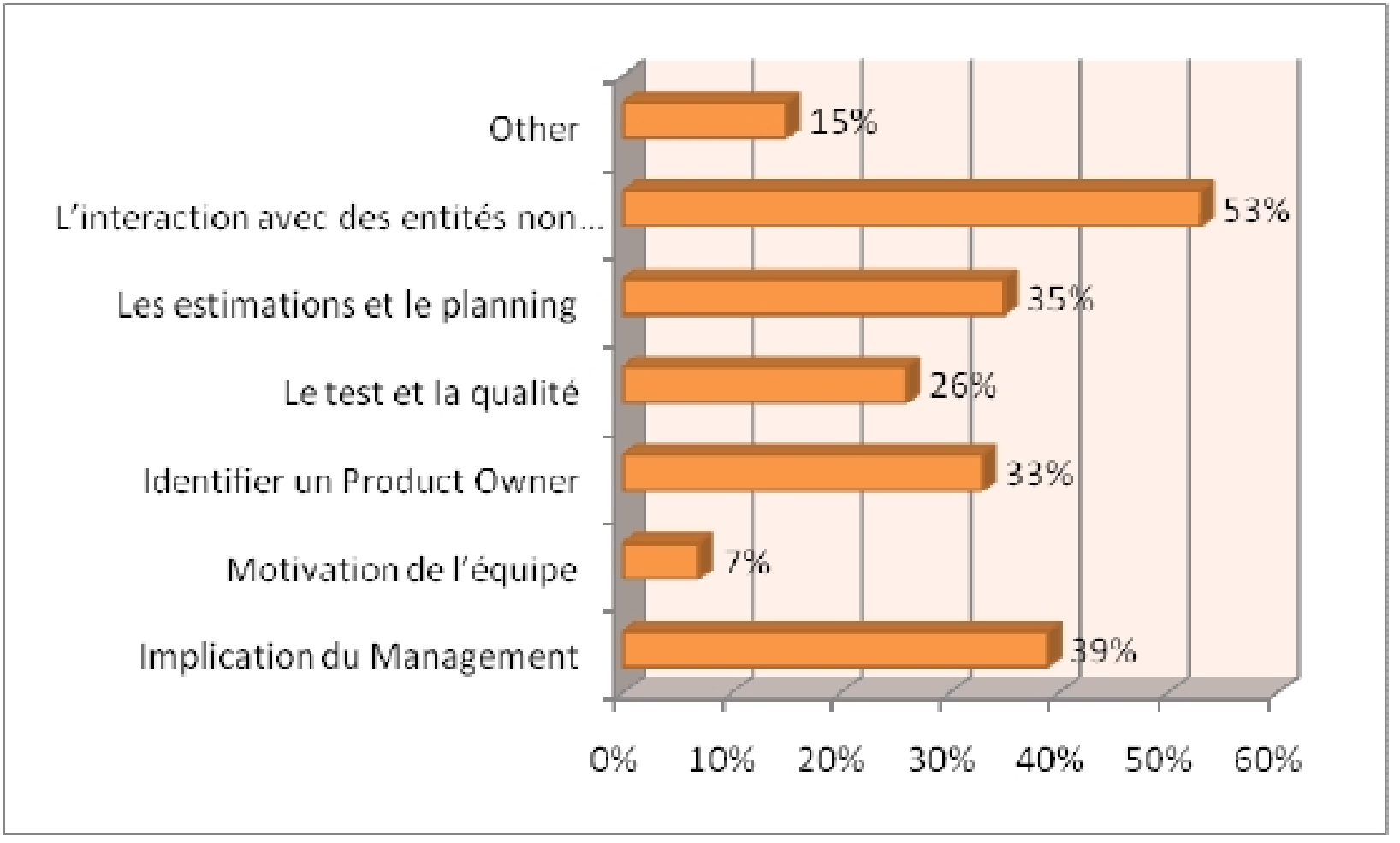
C'est toujours plus simple de démarrer avec un nouveau projet (que de reprendre un projet avec des millions de lignes de code et une base clients énorme)

Scrum apporte motivation et implication à l'équipe (si ce n'est pas le cas cela le ScrumMaster devrait s'alerter)



Se réferer à l'étude : Enquête Nationale Méthodes Agiles Juin 2009
http://www.frenchsug.org/download/attachments/591296/Enquête_méthodes_agiles_2009_FrenchSUG.pdf?version=1

Gérer le changement : Difficulté



Se réferer à l'étude : Enquête Nationale Méthodes Agiles Juin 2009
http://www.frenchsug.org/download/attachments/591296/Enquête_méthodes_agiles_2009_FrenchSUG.pdf?version=1

Face au stress

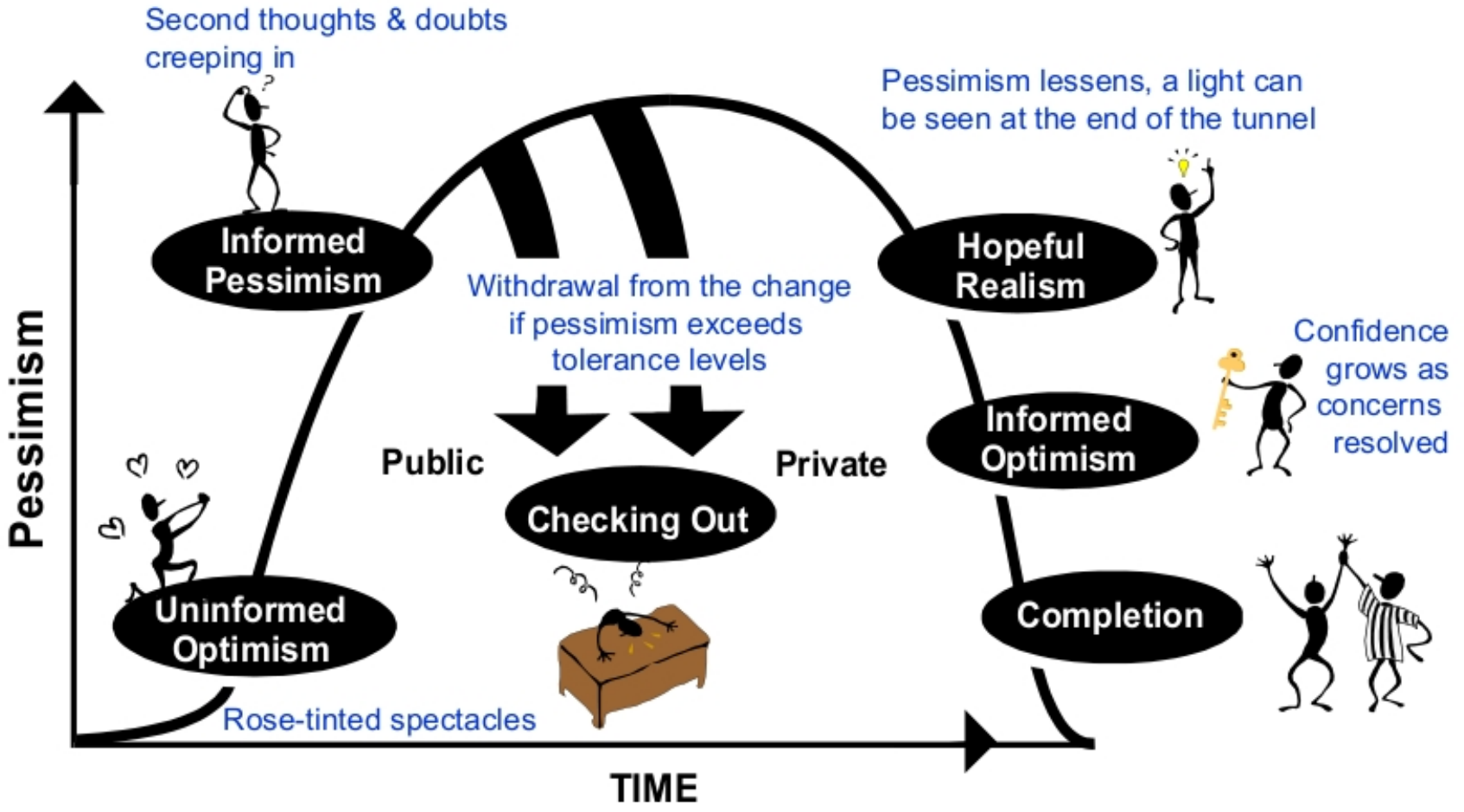
Dès que le stress va monter les bonnes pratiques et les bonnes volontés vont disparaître et les mauvaises habitudes revenir au galop

Dans des phases de stress vous devez être attentif à ne pas :

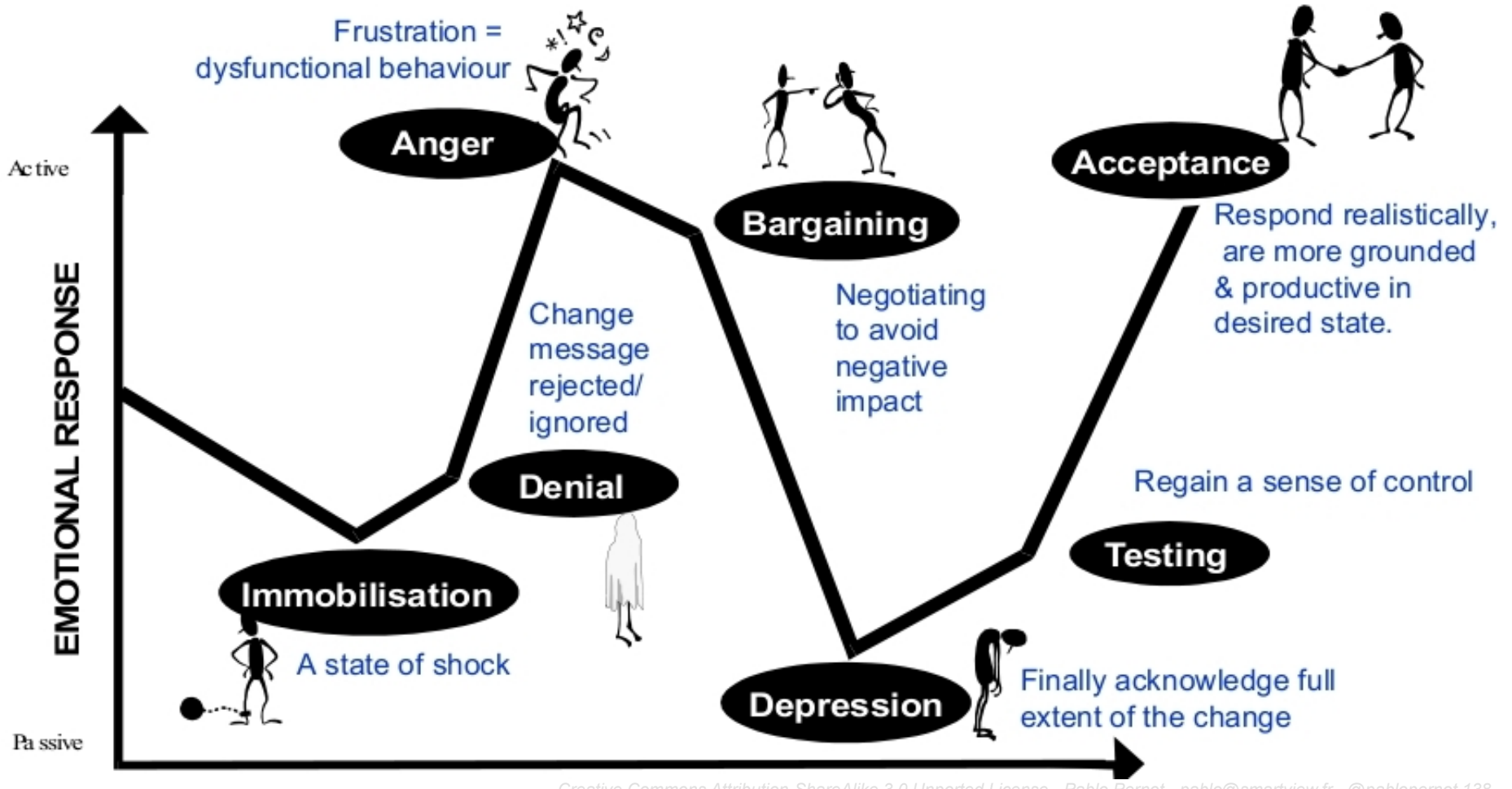
- Retomber dans un schéma de type Waterfall
- Retomber dans un schéma de type « commande et contrôle » (rappelez vous l'équipe est autonome et autogérée).
- Se voiler la face et dire oui à ce qui est impossible à réaliser dans les conditions proposées (durée, équipe, etc.)
- Cacher la réalité en se disant que cela va se résoudre

La qualité ne doit jamais être remise en cause (« j'arrête la documentation », « j'arrête les tests pour gagner du temps »).

Positive Response to Change



Negative Response to Change



CONTRACTUALISATION

Scrum fonctionne généralement avec des contrats de type « régie » (time & material)

La difficulté est d'adapter les habitudes françaises du mode de contractualisation au forfait (fixed price) à la souplesse proposée par Scrum.

C'est encore plus dur avec les appels d'offres publics

C'est souvent un risque que l'organisation décide de prendre (de gérer le forfait avec SCRUM)

Mode dégradé : sans pouvoir se permettre de modifier le périmètre mais en exploitant la priorisation, les rétrospectives fréquentent auprès du client

Mode très dégradé : uniquement la partie réalisation sans nécessairement que le client soit au courant (mais ce n'est plus vraiment du Scrum)

Le point de vue du juriste

http://www.staub-associes.com/fr/page-5-286-methodes-agiles.html

Contractualisation : contrat avec changement gratuit

Jeff Sutherland, l'un des pères de Scrum, propose le concept de « changement gratuit »

On définit un contrat avec un périmètre et un prix fixe (forfait / fixed price).

On propose une enveloppe supplémentaire en mode régie (time & material) qui sera déclenchée si la clause de changement gratuit échoue.

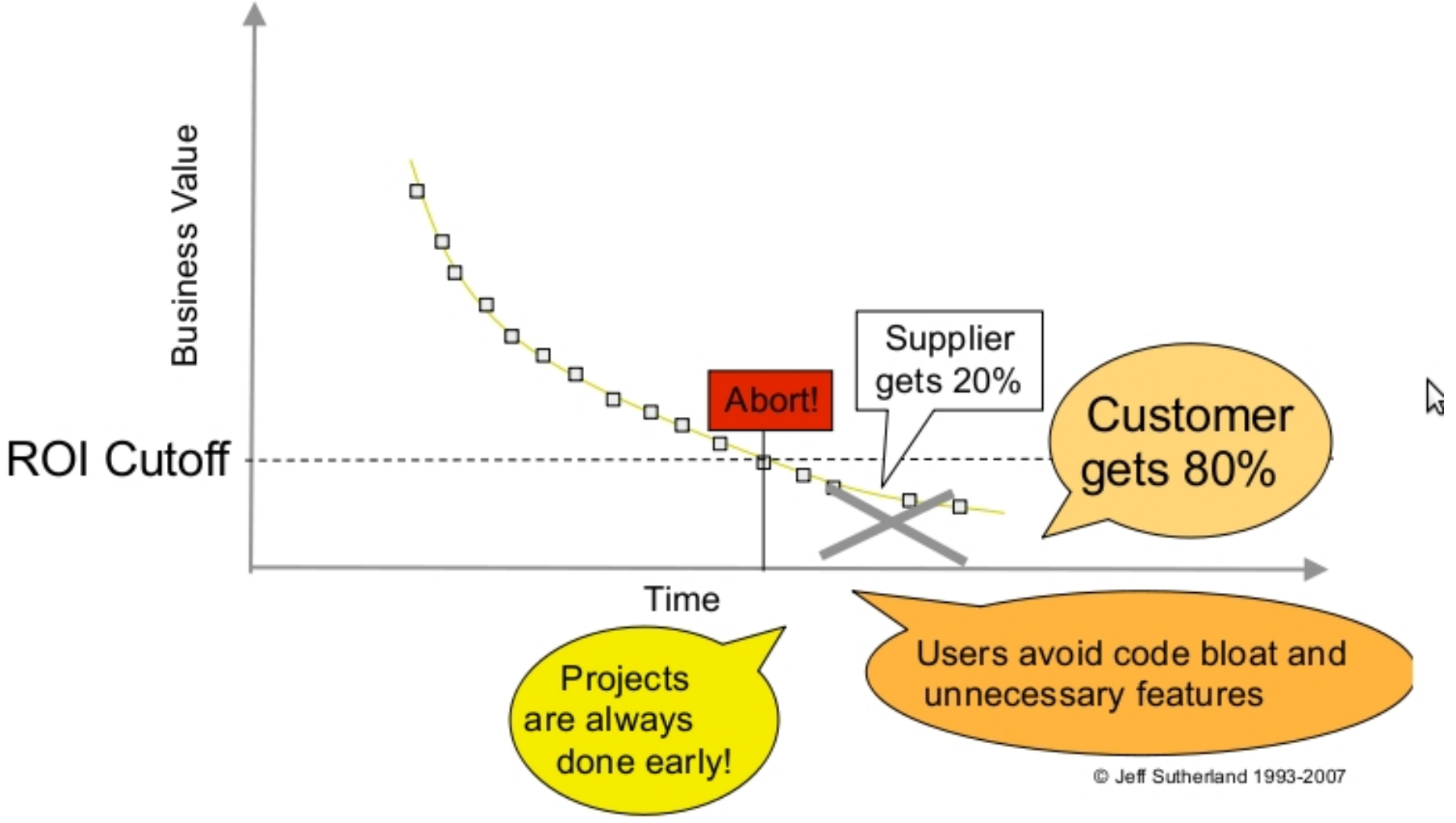
On introduit une clause de changement gratuit

Pour que cette clause soit valide le client doit s'engager à travailler avec l'équipe Scrum (Product Owner ou autres). Si il ne respecte pas cette clause l'enveloppe supplémentaire sera utilisée en cas de variation.

Si il respecte cette règle le client peut effectuer des changements dans le périmètre :
- Le changement des priorités est possible
- Il peut enlever des « features » et en ajouter de nouvelles (estimation égale) à la fin de chaque sprint.

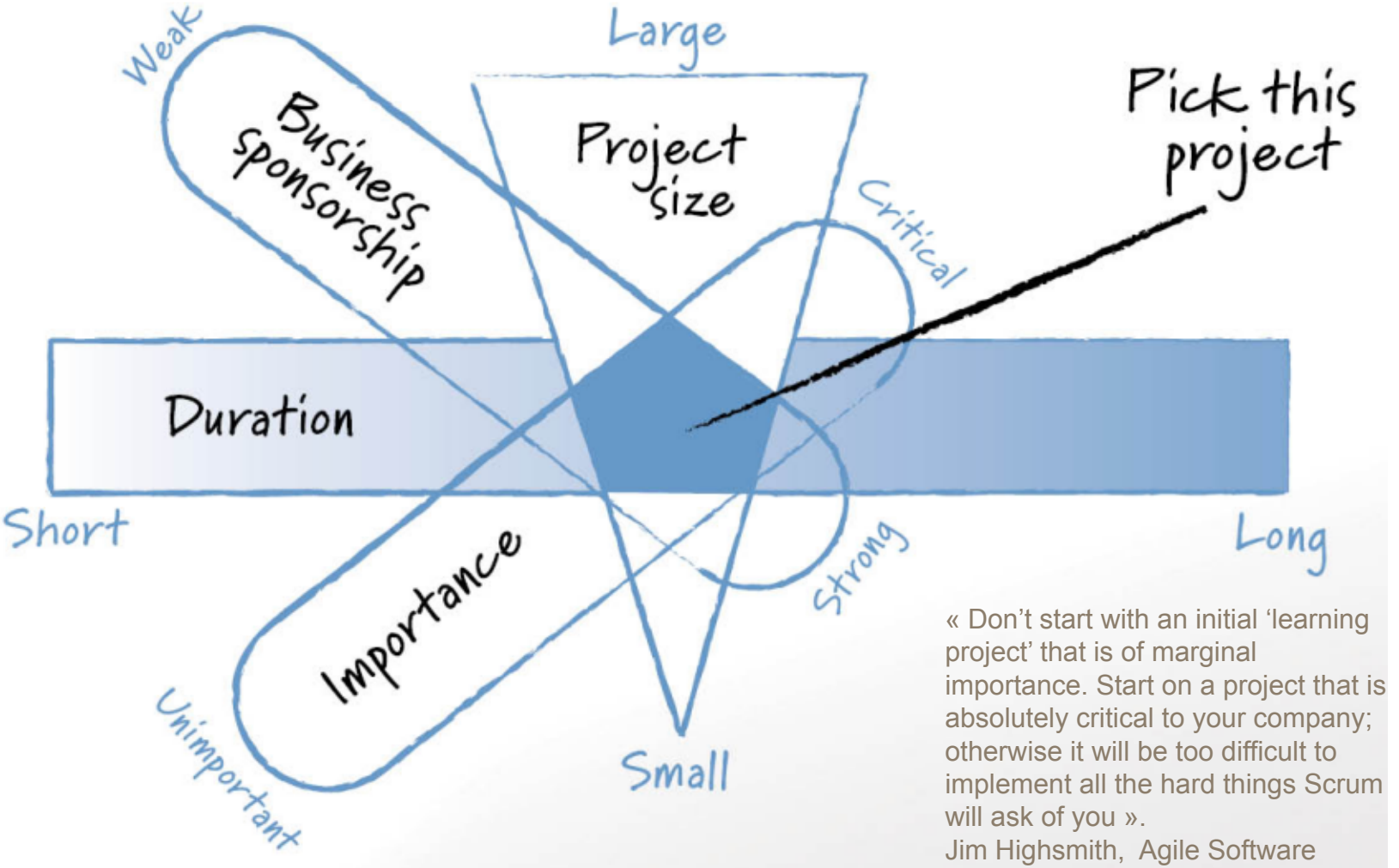
Source : http://agile2008toronto.pbworks.com/Money-For-Nothing-And-Your-Changes-For-Free-(Jeff-Sutherland)

MONEY FOR NOTHING



Source : http://agile2008toronto.pbworks.com/Money-For-Nothing-And-Your-Changes-For-Free-(Jeff-Sutherland)

Le projet pilote ?



« Don't start with an initial 'learning project' that is of marginal importance. Start on a project that is absolutely critical to your company; otherwise it will be too difficult to implement all the hard things Scrum will ask of you ».
Jim Highsmith, Agile Software Development Ecosystems.

Source : Mountain Goat Software
http://blog.mountaingoatsoftware.com/four-attributes-of-the-ideal-pilot-project

Management agile

Acteur du changement, initiateur, constructeur de l'organisation

Mène les questions de recrutement et de rétribution. Fournit les ressources.

Responsable des Product Owner (Donne son avis au Product Owner concernant la stratégie et la vision du produit, et donne son avis au Product Owner sur l'organisation et l'orientation du Product Backlog)

Vient aider les équipes pour supprimer les obstacles qui pourraient les gêner.

Aide les ScrumMasters à protéger les équipes des perturbations extérieures

Il aide l'équipe si elle a un besoin ponctuel (support technique)

Coordinateur (entre différentes strates de la société)

Source : Le rôle du manager dans Scrum, Agile Gathering 2007
Traduit par Fabrice Aimetti
http://www.fabrice-aimetti.fr/dotclear/public/mes-documents/Role-des-Managers-En-Scrum.pdf

Management agile

« the Team will not begin to self-organize until everyone outside the Team stops micromanaging them."
Pete Deemer, Manager 2.0: The Role of the Manager in Scrum
http://www.scrumalliance.org/articles/148-manager--the-role-of-the-manager-in-scrum

Il ne doit plus y avoir de vision « top / down ». Le manager doit constamment rappeler à l'équipe qu'elle est le responsable (chassez la naturel il revient au galop ! Le « muscle memory » de Ken Schwaber)

Le manager est plus un « mentor » qu'une « nounou ».

La gestion des individualités a disparu. Il faut revoir le plan de commissionnement au niveau de l'équipe et du projet !

Source : Le rôle du manager dans Scrum, Agile Gathering 2007
Traduit par Fabrice Aimetti
http://www.fabrice-aimetti.fr/dotclear/public/mes-documents/Role-des-Managers-En-Scrum.pdf

Atelier "Leadership"

atelier
"Leadership"

Responsabilité

Christopher Avery Responsability Process

http://www.christopheravery.com/responsibility-process

Scalabilité : Scrum de scrum

Permet de répondre aux problématiques des projets avec plusieurs équipes.

Chaque jour après le Daily Scrum le représentant de chaque équipe s'assemble pour un « scrum de scrum ».

Cela fonctionne comme un daily scrum mais avec un niveau d'analyse différent : qu'a fait l'équipe, que fait-elle aujourd'hui, quels problèmes rencontre-t-elle ? (Mike Cohn suggère des meetings moins fréquents mais plus long)

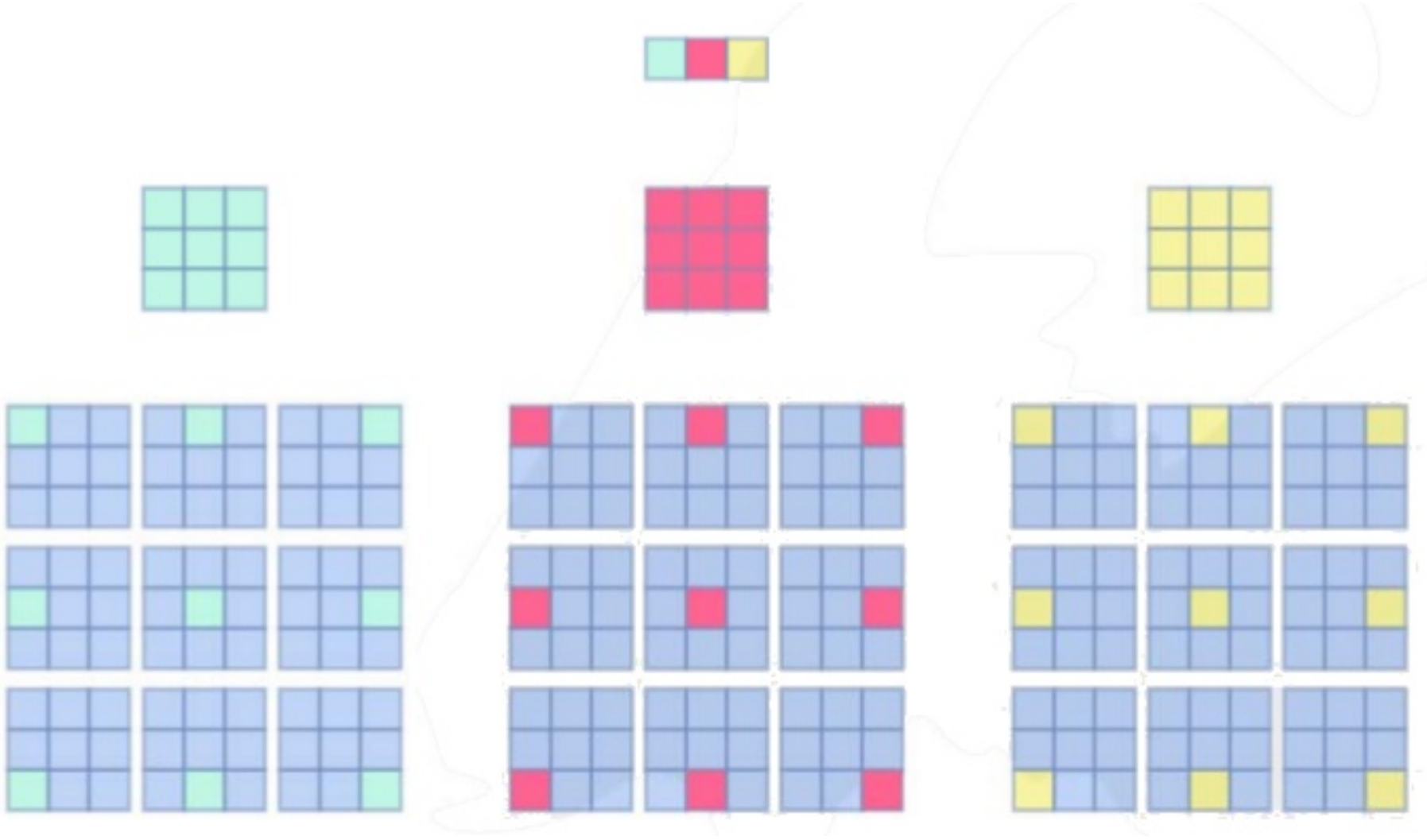
Le risque est que le représentant de chaque équipe ne soit pas en mesure de bien décrire l'activité de son équipe (mémoire, précision, etc.)

Ken Schwaber préconise l'obligation à toutes les équipes de fournir un résultat lors des revues de sprint et de l'intégrer de façon à ce que le Product Owner puisse valider convenablement le résultat.

L'idée derrière les scrums de scrum est de décentraliser chaque unité de productivité de façon à ce qu'elle s'auto-organise. Mais de recomposer une action globale au travers de la méthode agile : inspection, adaptation, transparence.

Source : ken Schwaber, Mike Cohn
http://kenschwaber.wordpress.com/2010/07/27/the-evolution-and-importance-of-the-scrum-of-scrums/

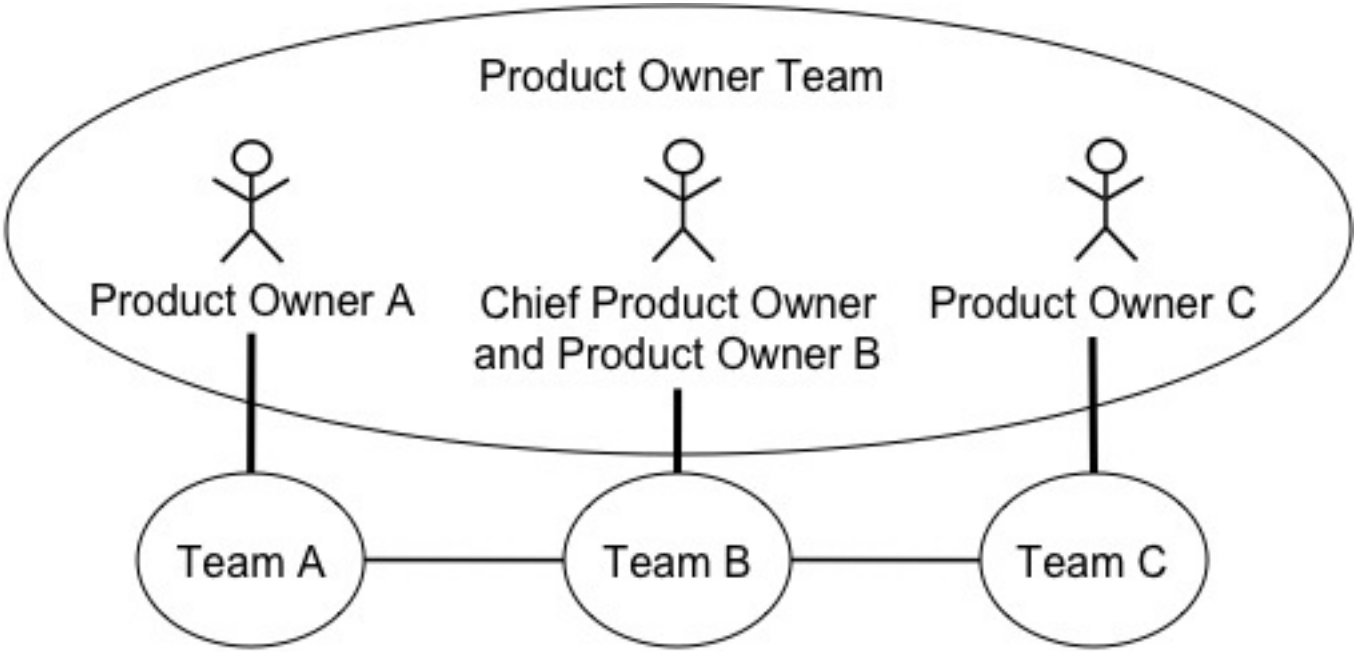
Scalabilité : Scrum de scrum



Source : Mountain Goat Software
http://www.mountaingoatsoftware.com/system/presentation/file/30/RedistributableIntroToScrum.ppt

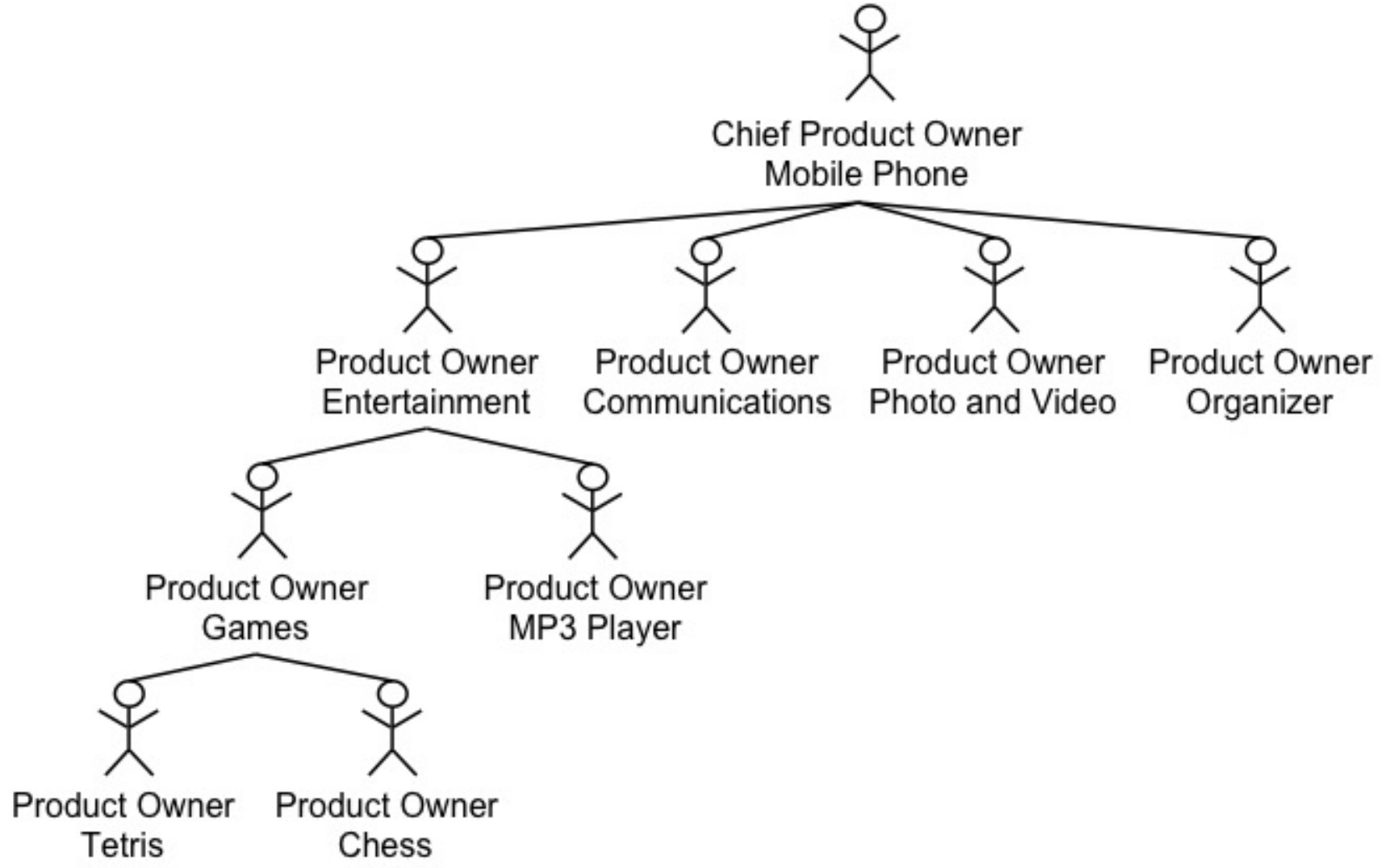
Product owner dans l'entreprise

Comme les "scrum de scrums", les product owners peuvent se synchroniser régulièrement pour aborder les questions de stratégie plus globale.



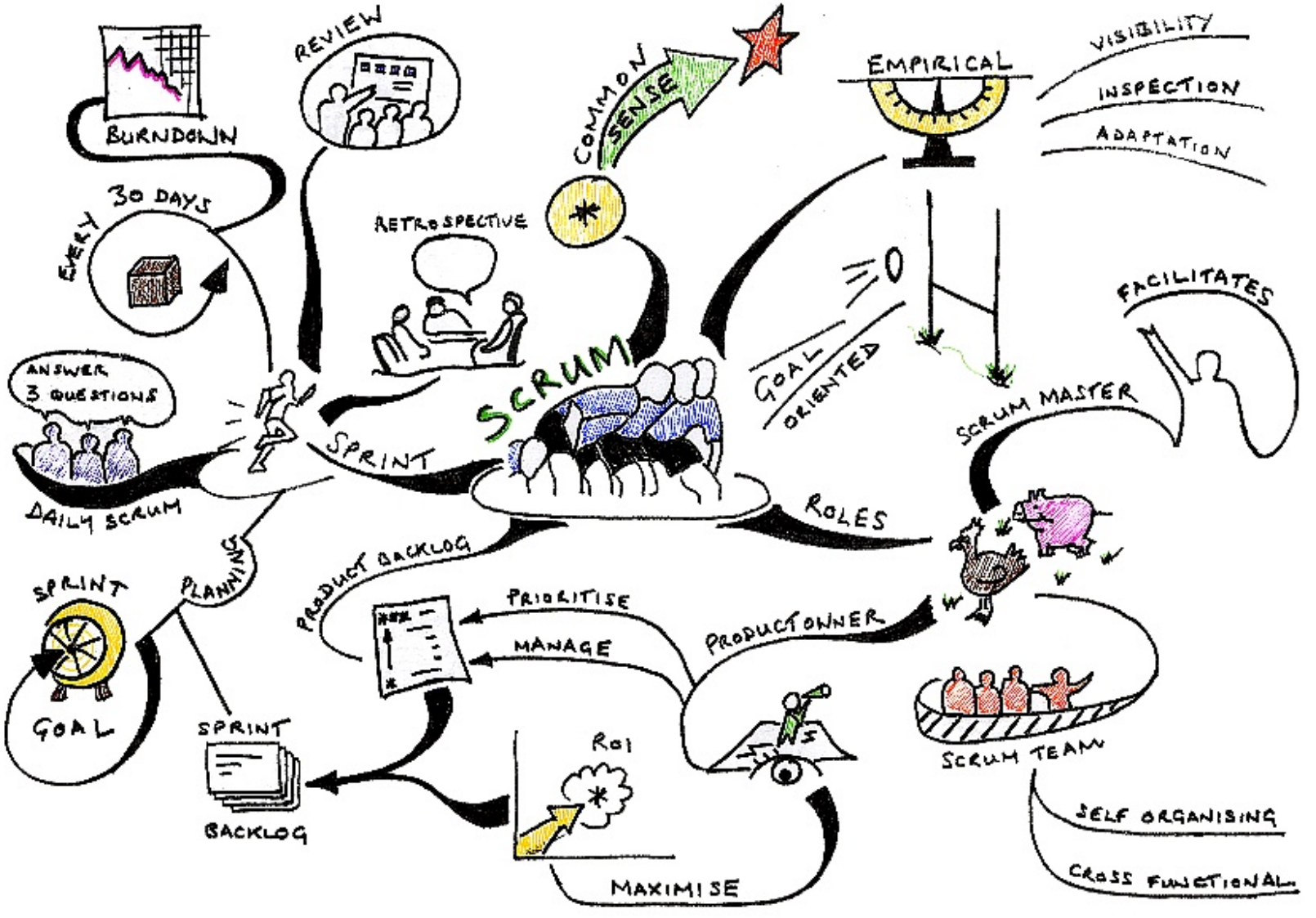
Source :http://www.romanpichler.com/blog/roles/scaling-the-product-owner/

Product owner dans l'entreprise



Source :http://www.romanpichler.com/blog/roles/scaling-the-product-owner/

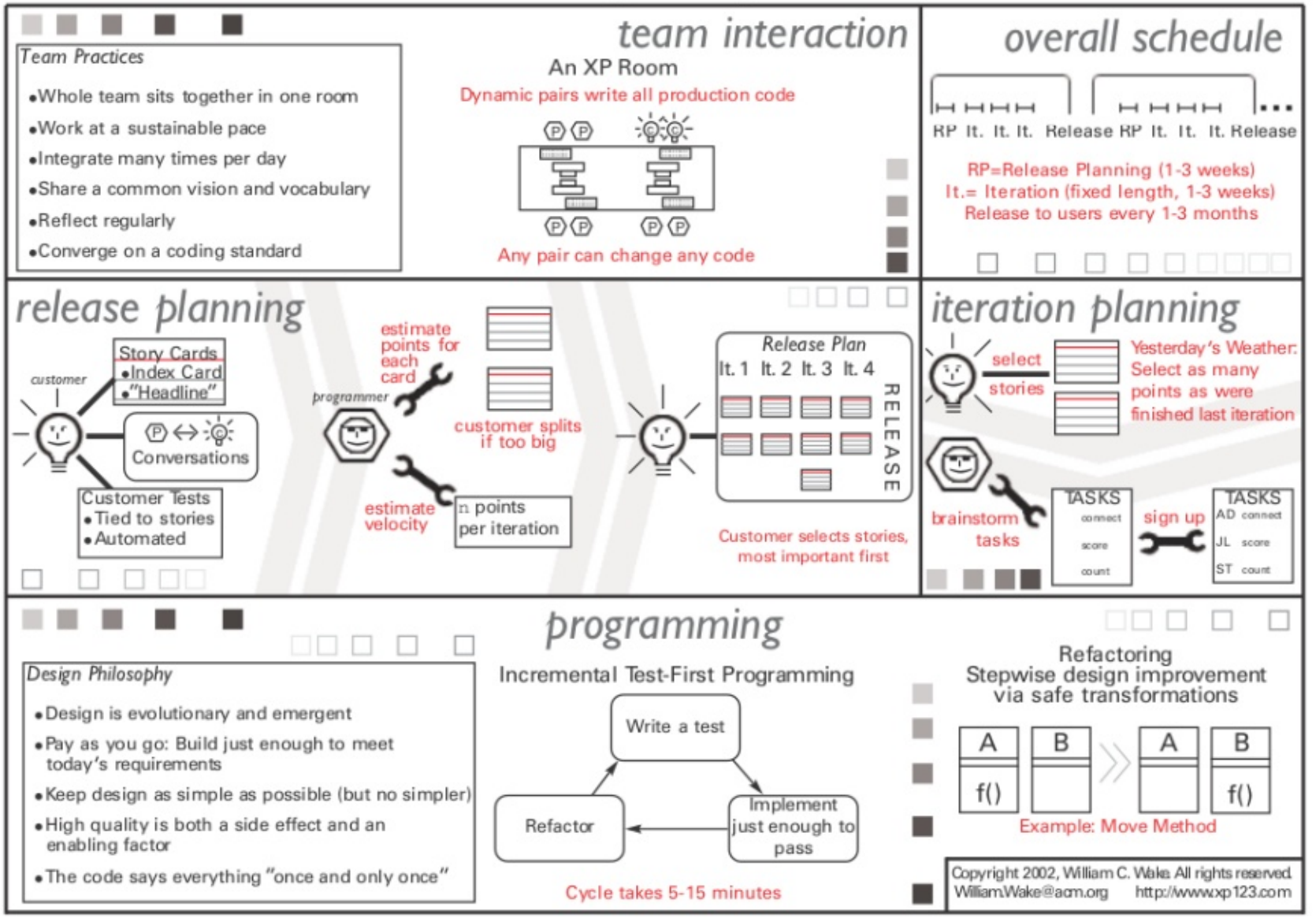
Quelques rappels : Scrum Mind Map



http://www.fabrice-aimetti.fr/dotclear/index.php?post/2009/07/12/Small-and-Funny-Scrum-Mindmap

Droits : http://www.flickr.com/photos/agileinaction/ / CC BY 2.0

QUELQUES RAPPELS : XP SUR UNE PAGE



Source : http://xp123.com/xplor/xp0202/

Quelques rappels : Points de vigilance

Stabilité de l'équipe

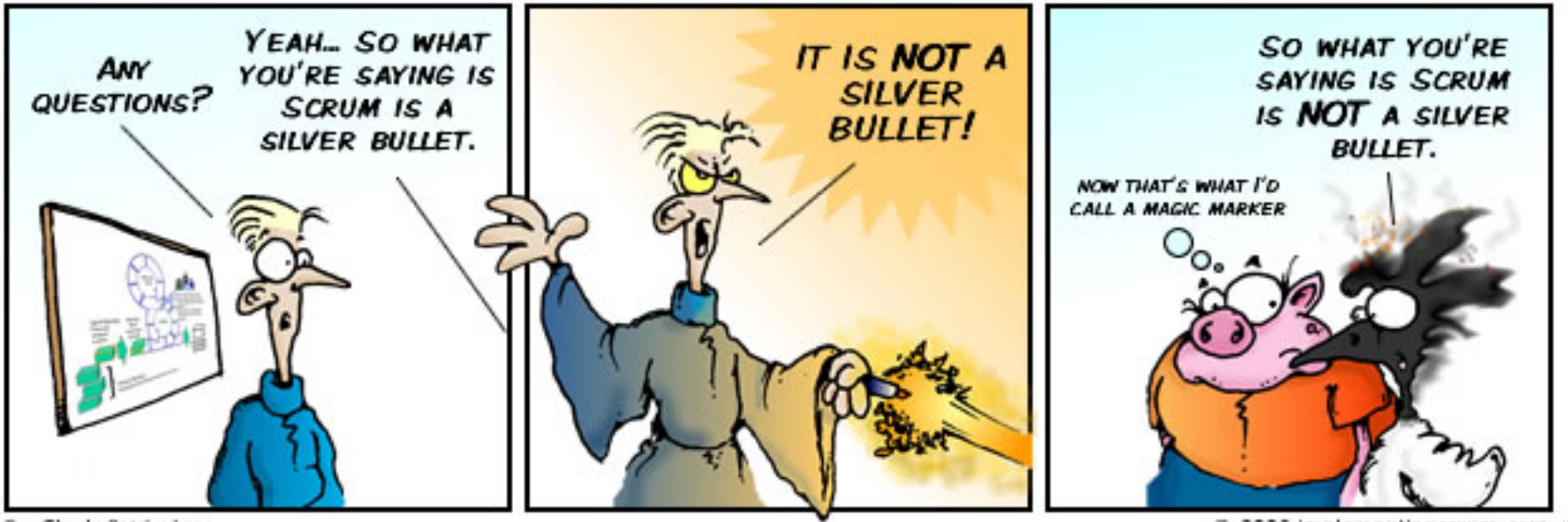
Protection de l'équipe vis à vis des parties prenantes (stakeholders)

Contractualisation

Responsabilisation du product owner et de l'équipe de développement

Plus de chef de projet mais un ScrumMaster (facilitateur)

L'agile est simple à appréhender mais il n'est pas simple à déployer



Ressources : Liens

http://www.scrum.org/ EN

http://agileanarchy.wordpress.com/ (Tobias Mayer) EN

http://agilmanifesto.org EN

http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf EN

http://www.crisp.se/henrik.kniberg/ScrumAndXpFromTheTrenches.pdf ***EN***

http://www.mountaingoatsoftware.com (Mike Cohn) ***EN***

http://leanovation.org/ & http://oanasagile.blogspot.fr/ (Oana Juncu) [Convergences !]

http://www.aubryconseil.com/ (Claude Aubry) FR

http://thierrycros.net/ (Thierry Cros) FR

http://blog.avoustin.com/ (Jérôme Avoustin) FR [SmartView !]

http://www.youtube.com/watch?v=IyNPeTn8fpo (Ken Schwaber Scrum Google Talk) ***EN***

http://flowchainsensei.wordpress.com/ (Bob Marshall) ***EN***

http://xp123.com ***EN***

http://xprogramming.com/ ***EN***

Ressources : Liens

http://blog.crisp.se/henrikkniberg/2011/02/17/1297928460000.html (Lean from the trenches)

http://agilemanagement.net/ (David Anderson)

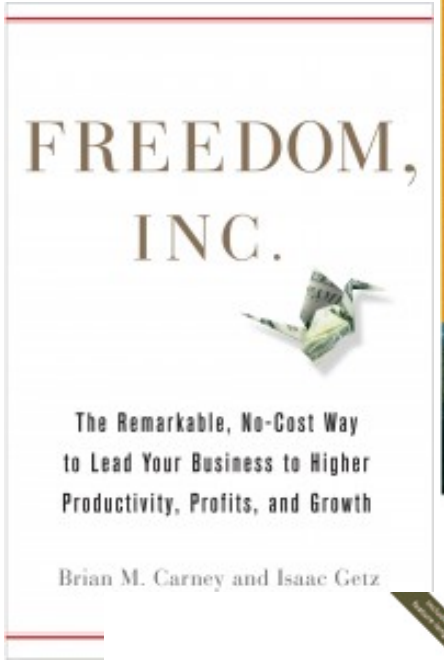
http://martinfowler.com/ (Martin Fowler)

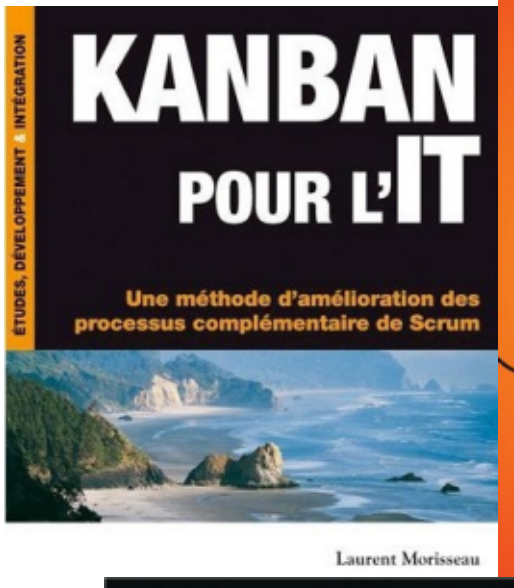
http://www.exampler.com/ (Brian Marick)

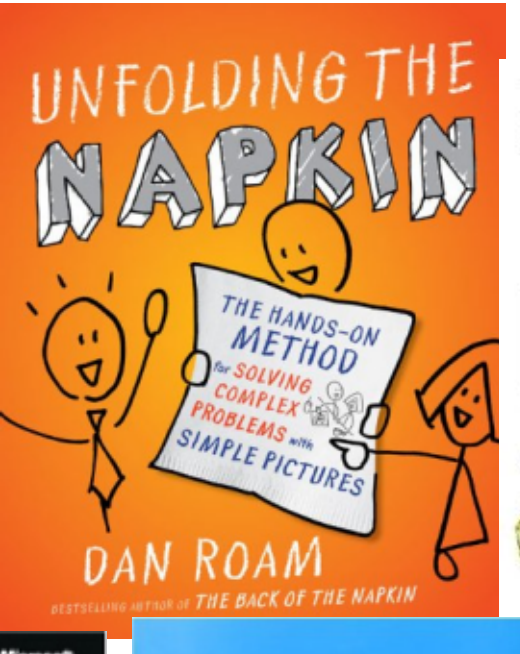
http://www.jimhighsmith.com/ (Jim Highsmith)

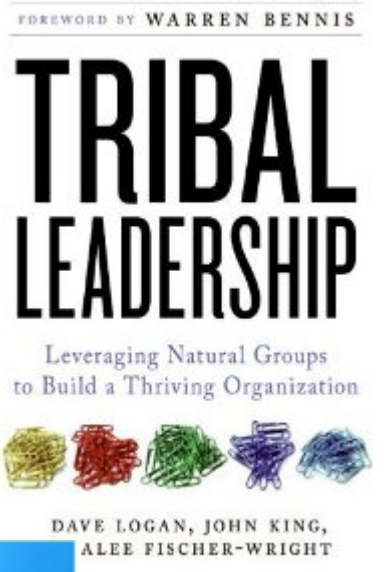
Ressources : Livres

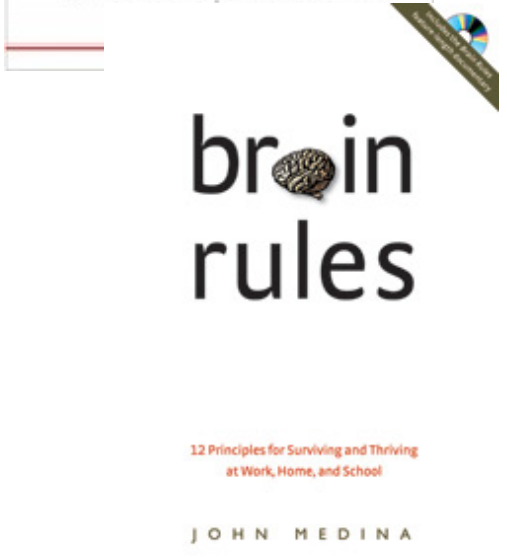
Ressources : Livres

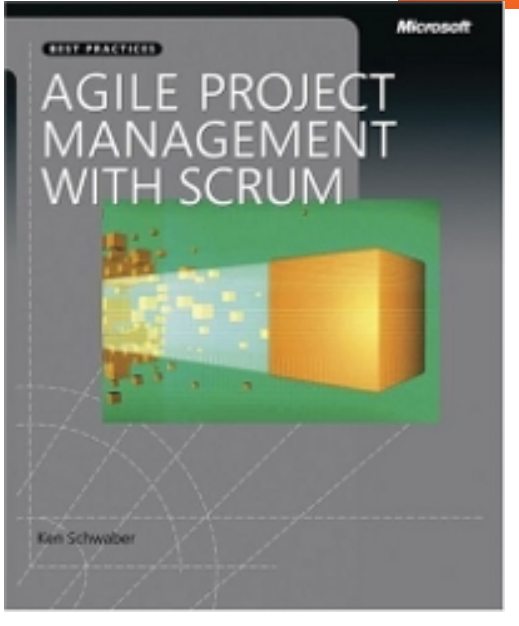


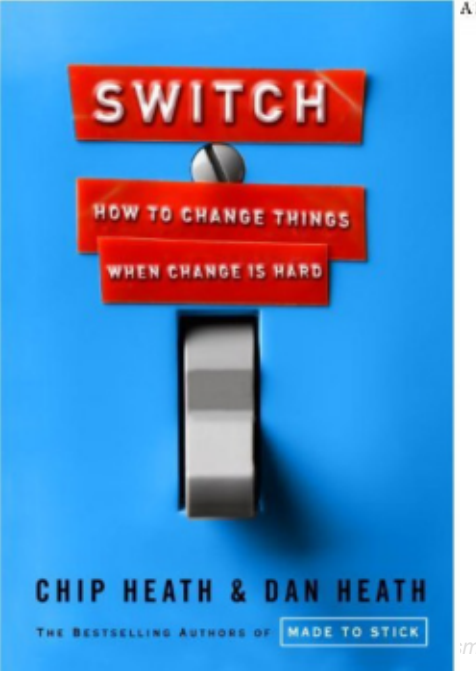












Ressources : Outils

Rappelez vous que rien ne vaut l'interaction, la visibilité, etc.
Le mur, les post-it, la patafix, les tableaux blancs seront toujours vos meilleurs outils.

Les outils agiles devraient utilisés :

- Pour consolider les informations
- Pour éventuellement servir au sein d'équipes distribuées

Outils

Redmine, Trac

Icescrum, Kunagi

Mingle (Thought works)

Rally software, Version One, Pivotal Tracker

Jira/Greenhopper (Atlassian)

Agile Zen, Kanbanery

QUESTIONS / RÉPONSES

PABLO PERNOT - SMARTVIEW

http://www.areyouagile.com
https://speakerdeck.com/u/pablopernot
http://www.smartview.fr
http://convergenc.es

@pablopernot