Introduction aux Méthodes Agiles

Rémy BROUSSET
remy.brousset@aquilenet.fr

PLAN DU COURS

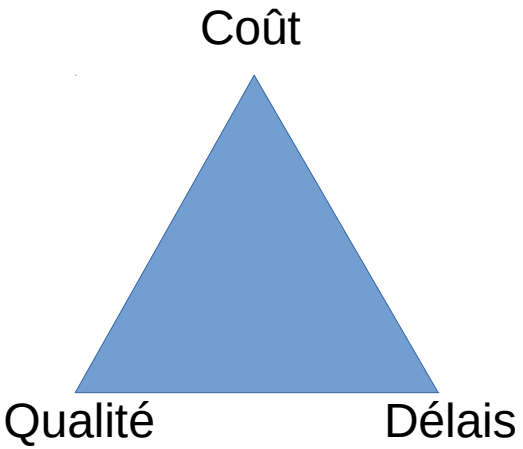
- Gestion de projet « classique »
  - Particularités des projets IT
- Principes et concepts des méthodes agiles
  - Principes LEAN
- Approche agile vs Approche « classique »
- Méthode Scrum
- Méthode Kanban
- Méthode eXtreme Programming

Rappels

Projet : ensemble finalisé d'activités et d'actions coordonnées dans le but de répondre à un besoin défini dans des délais fixés (date de début et date de fin), selon l'enveloppe budgétaire et les ressources allouées, et produisant des livrables.

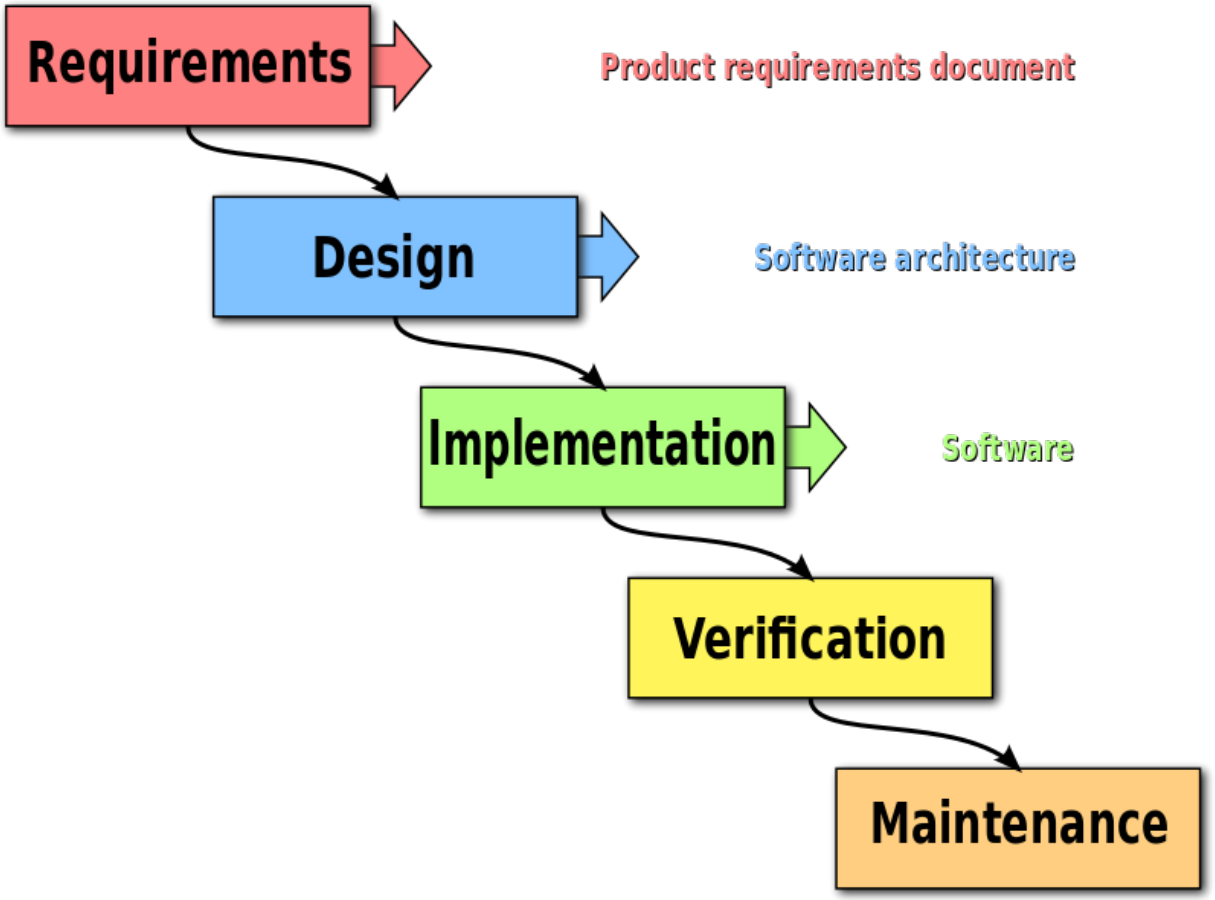
Rappels

Triangle Coût – Qualité - Délais

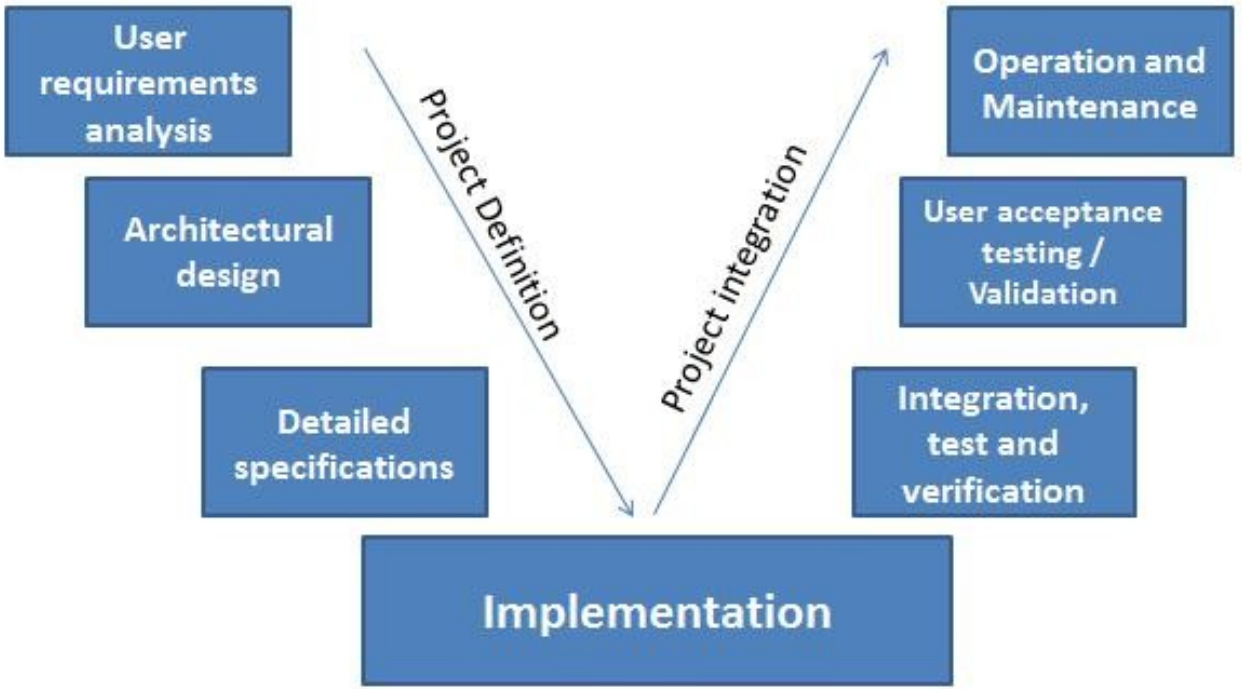


Pick two !!

Gestion de Projet en Cascade



Gestion de projet en V



PMBoK

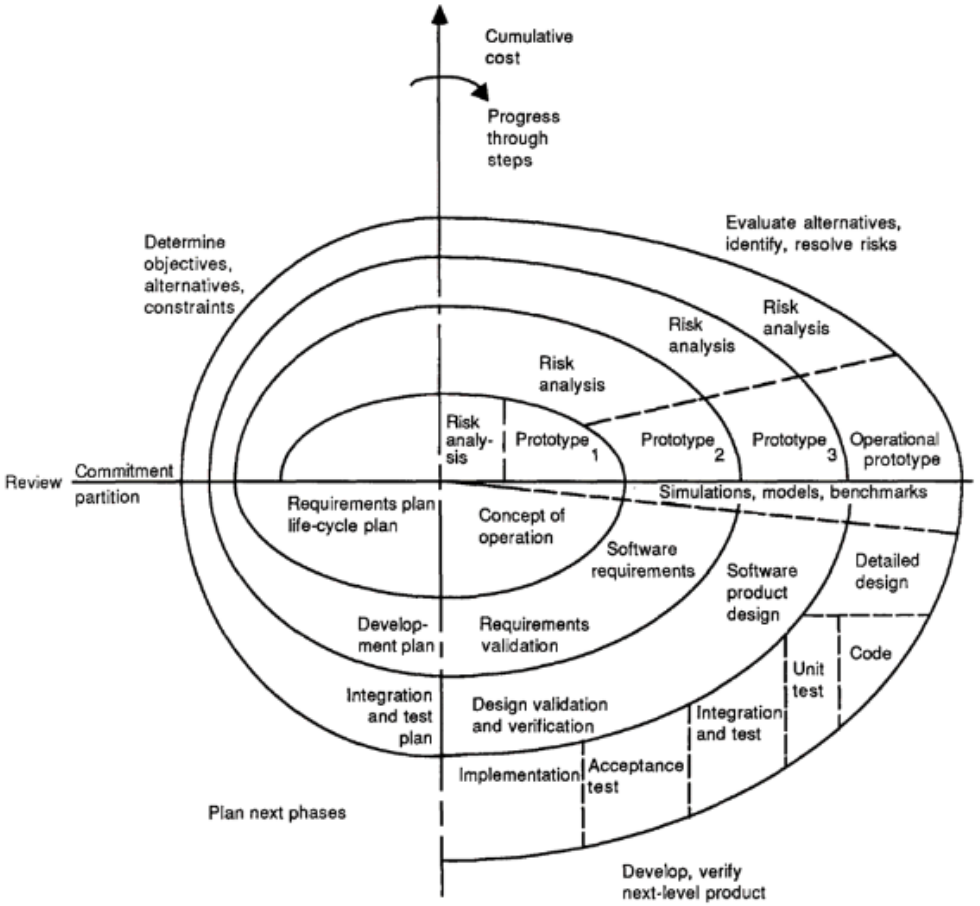
- PMI : Project Management Institute
- PMI Body of Knowledge
- Le PMBOK définit les 5 groupes de processus de management de projet suivants:
  - Démarrage
  - Planification
  - Exécution
  - Surveillance et maîtrise
  - Clôture

PMBoK

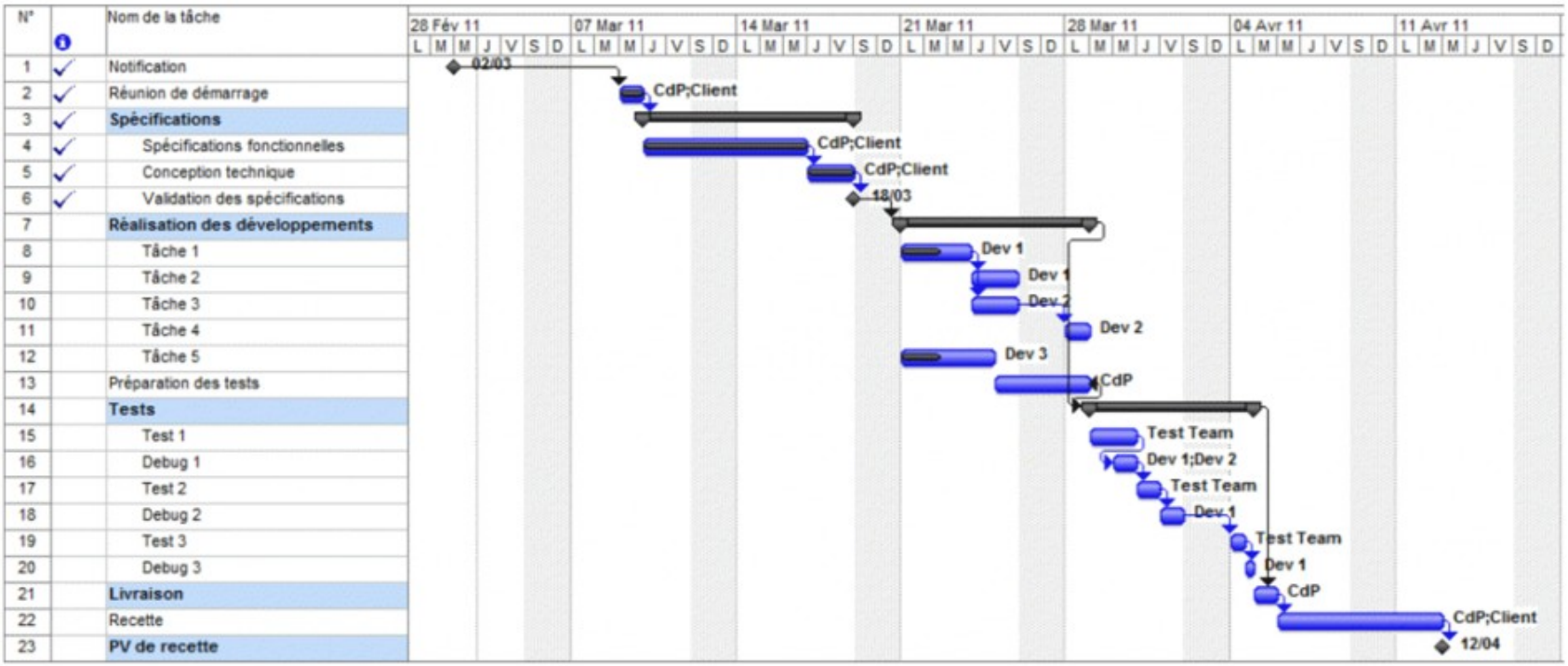
- Il aborde les dix domaines de compétences en management de projet, à savoir:
  - Le management de l'intégration
  - Le management du contenu (périmètre)
  - Le management des délais
  - Le management des coûts
  - Le management de la qualité
  - Le management des ressources humaines
  - Le management des communications
  - Le management des risques
  - Le management des approvisionnements
  - Le management des parties prenantes (stakeholders) ajouté dans la 5e édition

- Le PMBoK distribue un total de 49 processus dans ces groupes de processus et domaines de connaissance.

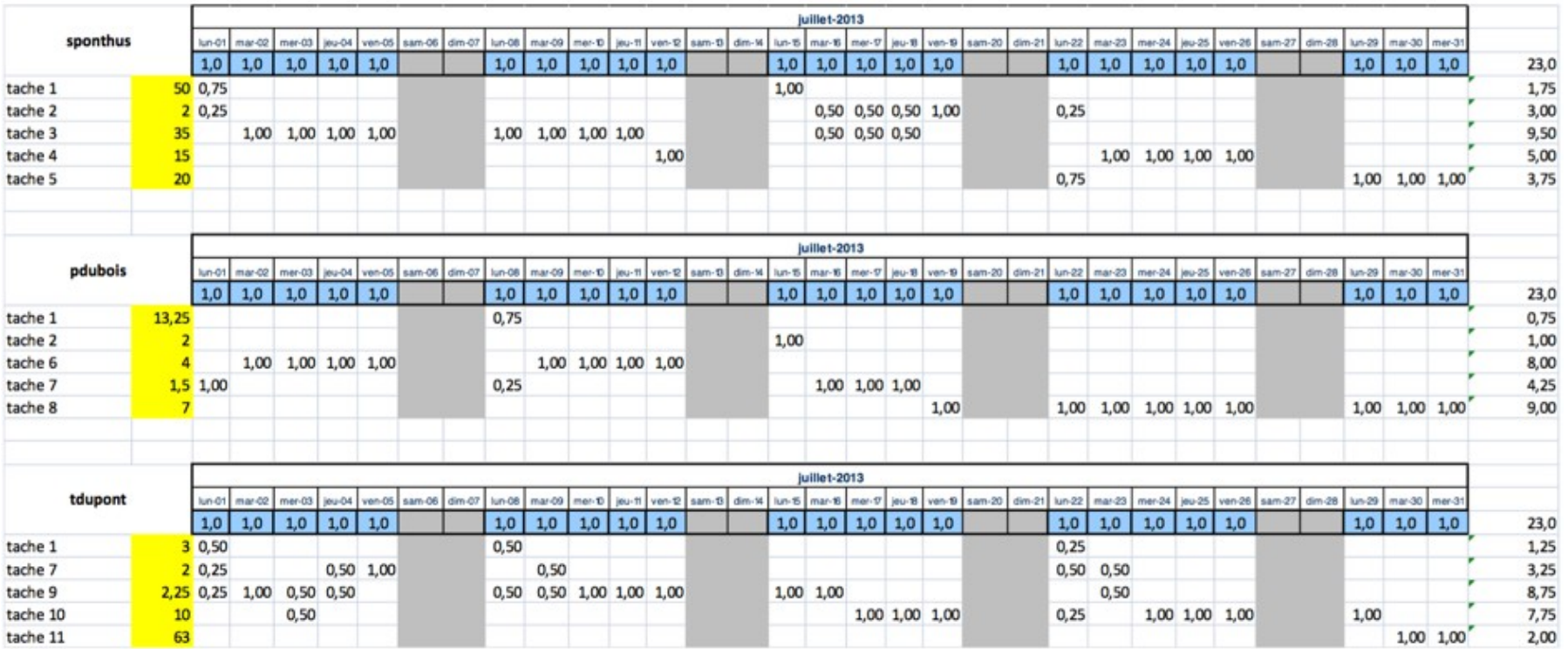
Gestion de projet en spirale



Planification



Planification



sponthus		juillet-2013																														
		lun-01	mar-02	mer-03	jeu-04	ven-05	sam-06	dim-07	lun-08	mar-09	mer-10	jeu-11	ven-12	sam-13	dim-14	lun-15	mar-16	mer-17	jeu-18	ven-19	sam-20	dim-21	lun-22	mar-23	mer-24	jeu-25	ven-26	sam-27	dim-28	lun-29	mar-30	mer-31	
		1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	23,0
tache 1	50	0,75														1,00																1,75
tache 2	2	0,25															0,50	0,50	0,50	1,00			0,25										3,00
tache 3	35		1,00	1,00	1,00	1,00			1,00	1,00	1,00	1,00					0,50	0,50	0,50														9,50
tache 4	15												1,00											1,00	1,00	1,00	1,00						5,00
tache 5	20																						0,75							1,00	1,00	1,00	3,75

pdubois		juillet-2013																														
		lun-01	mar-02	mer-03	jeu-04	ven-05	sam-06	dim-07	lun-08	mar-09	mer-10	jeu-11	ven-12	sam-13	dim-14	lun-15	mar-16	mer-17	jeu-18	ven-19	sam-20	dim-21	lun-22	mar-23	mer-24	jeu-25	ven-26	sam-27	dim-28	lun-29	mar-30	mer-31	
		1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	23,0
tache 1	13,25								0,75																							0,75
tache 2	2															1,00																1,00
tache 6	4		1,00	1,00	1,00	1,00				1,00	1,00	1,00	1,00																			8,00
tache 7	1,5	1,00							0,25								1,00	1,00	1,00														4,25
tache 8	7																			1,00			1,00	1,00	1,00	1,00	1,00			1,00	1,00	1,00	9,00

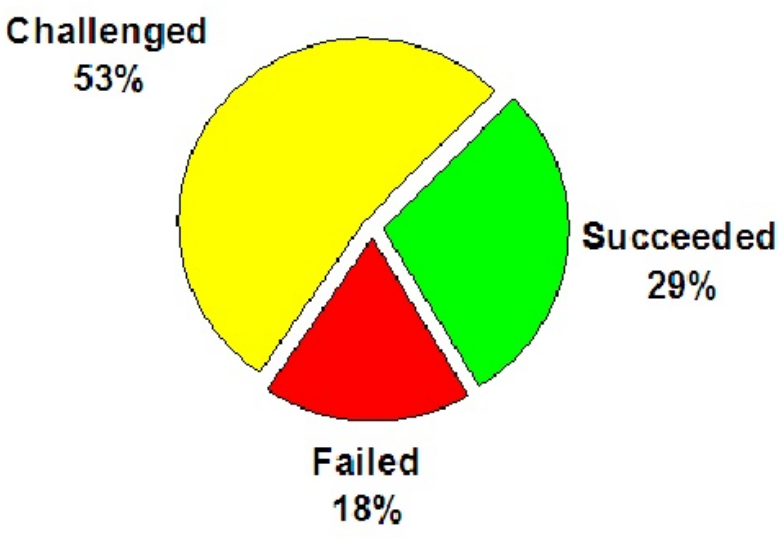
tdupont		juillet-2013																														
		lun-01	mar-02	mer-03	jeu-04	ven-05	sam-06	dim-07	lun-08	mar-09	mer-10	jeu-11	ven-12	sam-13	dim-14	lun-15	mar-16	mer-17	jeu-18	ven-19	sam-20	dim-21	lun-22	mar-23	mer-24	jeu-25	ven-26	sam-27	dim-28	lun-29	mar-30	mer-31	
		1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	1,0	1,0			1,0	1,0	1,0	23,0
tache 1	3	0,50							0,50														0,25										1,25
tache 7	2	0,25			0,50	1,00				0,50													0,50	0,50									3,25
tache 9	2,25	0,25	1,00	0,50	0,50				0,50	0,50	1,00	1,00	1,00			1,00	1,00							0,50									8,75
tache 10	10			0,50														1,00	1,00	1,00			0,25		1,00	1,00	1,00			1,00			7,75
tache 11	63																														1,00	1,00	2,00

Facteurs de succès d’un projet

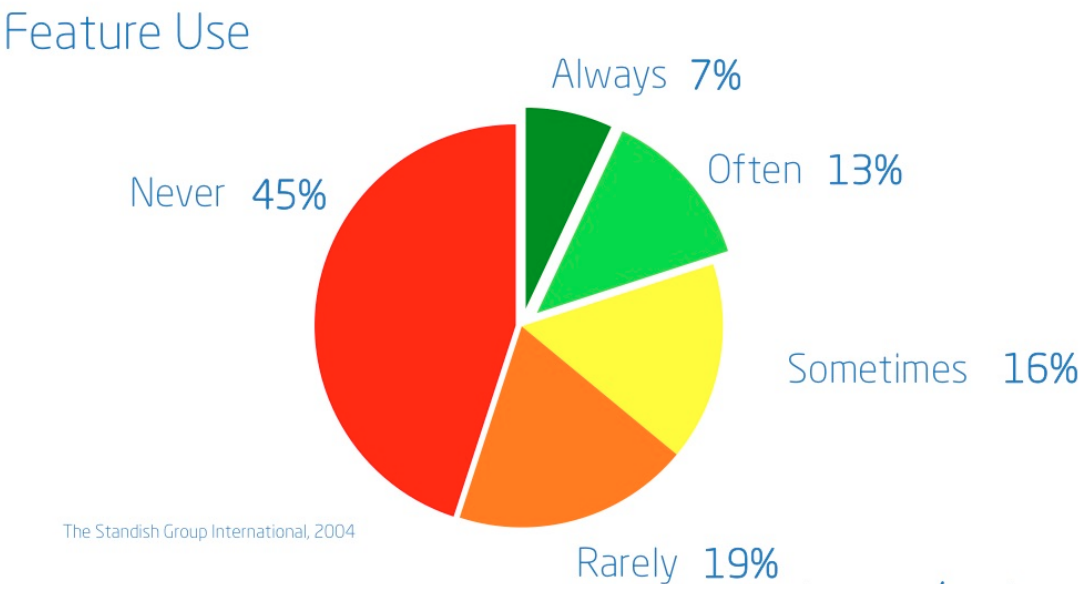
- Spécifications complètes
  - Difficile de détecter une fonctionnalité manquante
  - Fonctionnalité inutile
- Spécifications figées
  - Participation des utilisateurs
- Suivre le plan
  - Coût fixe
  - Dérives

Particularités des projets de génie logiciel

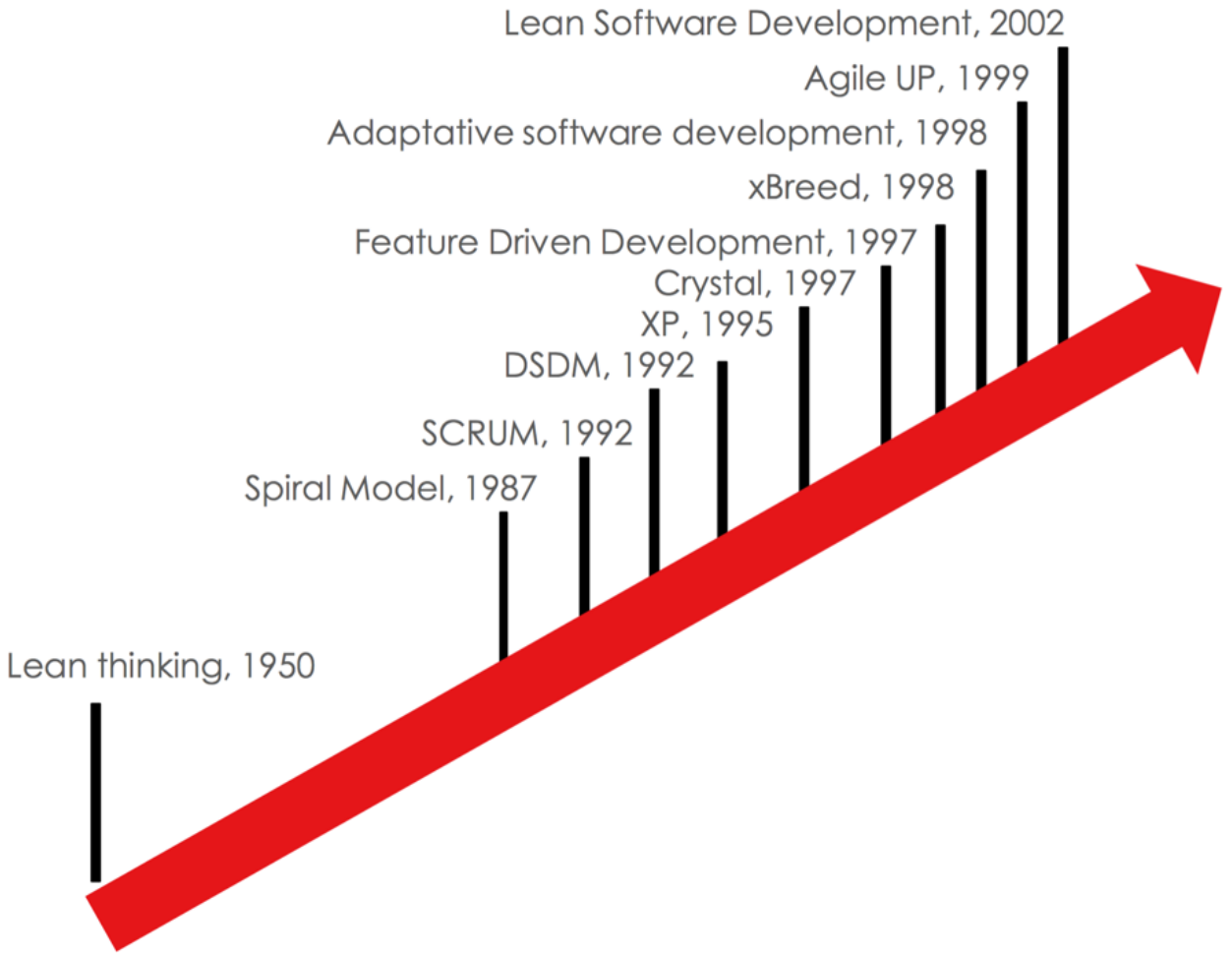
Immatériel → Unique → Innovant → Besoin incertain → Risqué



Copyright © 2006 The Standish Group International, Inc.

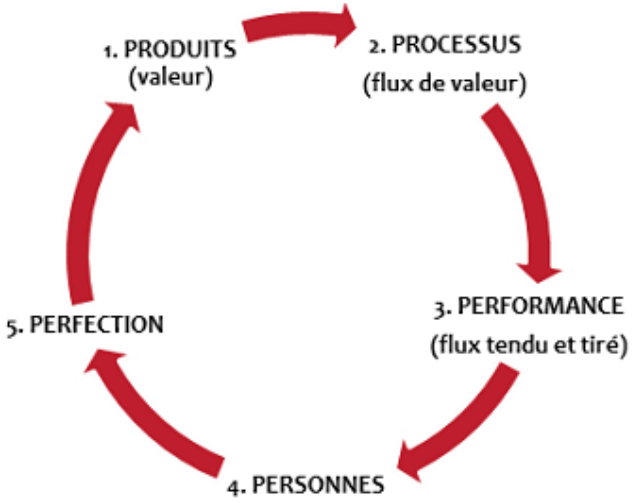


Apparition des méthodes



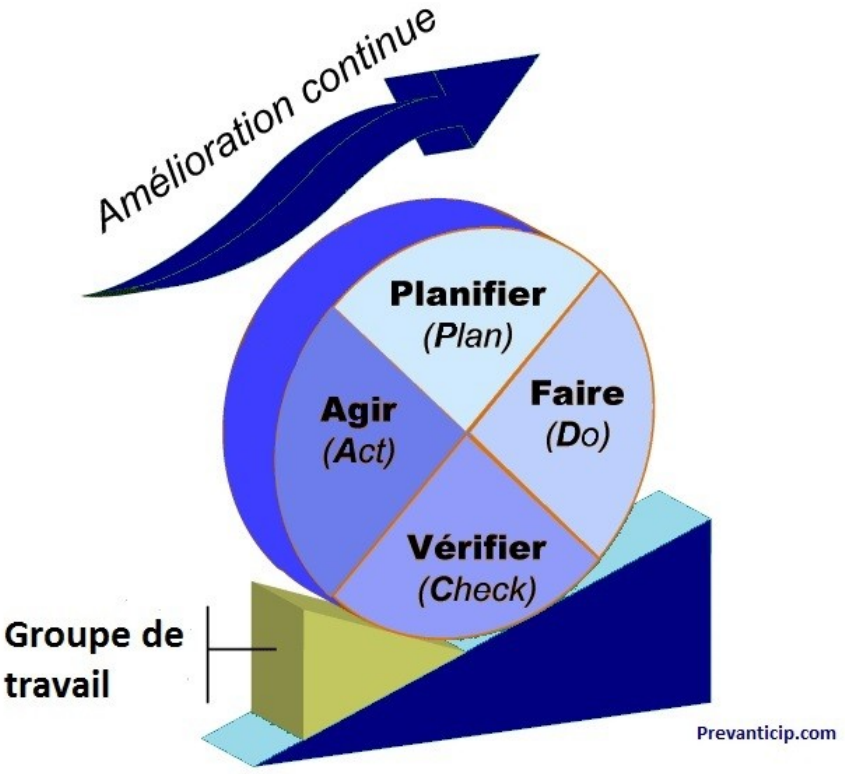
Principes LEAN

- Production de Valeur du produit
- Processus : Flux de Valeur
- Performance : Pull/JIT
- Personnes : Respect – Autonomie
- Perfection : Amélioration continue
  - Elimination des gaspillages (Kaïzen - Mud ,



Positionnement des méthodes	Niveau de management
Lean	Business management
Scrum	Project management
Extreme programming	Software engineering

La roue de Deming



Manifeste pour le Développement Agile de Solutions

Nous découvrons de meilleures façons de développer des solutions,
par notre propre pratique et en aidant les autres dans leur pratique.

Grâce à ce travail, nous en sommes venus à valoriser :

Les individus et leurs interactions, de préférence aux processus et aux outils,
Des solutions opérationnelles, de préférence à une documentation exhaustive,
La collaboration avec les clients, de préférence aux négociations contractuelles,
La réponse au changement, de préférence au respect d'un plan.

Précisément, même si les éléments à droite ont de la valeur,
nous reconnaissons davantage de valeur dans les éléments à gauche.

- https://manifesteagile.fr

Les Principes qui sous-tendent le Manifeste Agile

Nous respectons les principes suivants :

Notre principale priorité est de satisfaire le client en livrant rapidement et régulièrement des solutions qui apportent de la valeur.

Accueillez chaleureusement les changements de besoins, même tardifs dans le développement. Les processus agiles tirent parti du changement pour renforcer l'avantage concurrentiel du client.

Livrez souvent des solutions opérationnelles, à une fréquence allant de quelques semaines à quelques mois, avec une préférence pour les échelles de temps les plus courtes.

Les personnes en charge du métier ou des affaires et les personnes en charge de la réalisation doivent travailler ensemble chaque jour, tout au long du projet.

Construisez les projets à partir de personnes motivées. Donnez-leur l'environnement et le soutien dont elles ont besoin et faites-leur confiance pour mener à bien le travail.

La conversation en face à face est la méthode la plus efficace et la plus économique pour donner des informations à une équipe de réalisation, et pour échanger des informations à l'intérieur de l'équipe.

La disponibilité de solutions opérationnelles est la principale mesure d'avancement.

Les processus agiles encouragent à respecter un rythme soutenable lors de la réalisation. Les commanditaires, les réalisateurs et les utilisateurs devraient pouvoir maintenir indéfiniment un rythme constant.

Porter continuellement attention à l'excellence technique et à la qualité de la conception renforce l'agilité.

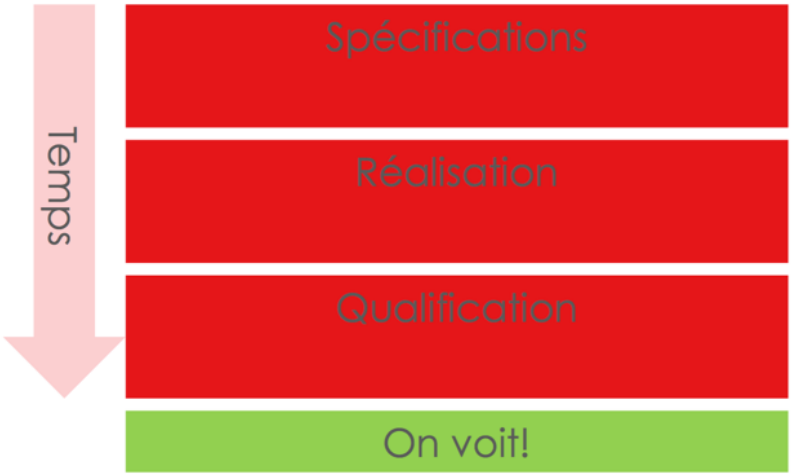
La simplicité – l'art de maximiser la quantité de travail qu'on ne fait pas – est essentielle.

Les meilleures architectures, les meilleures spécifications de besoins, et les meilleures conceptions émergent d'équipes auto-organisées.

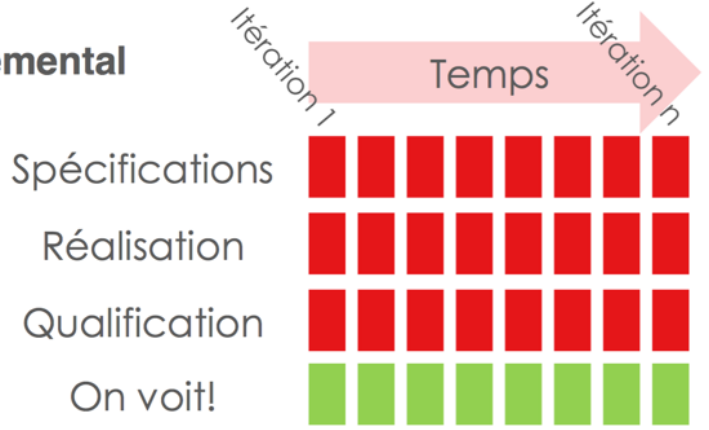
À intervalles réguliers, l'équipe réfléchit aux façons de devenir plus efficace, puis modifie son comportement et l'ajuste en conséquence.

Approche itérative

- **Cycle de vie linéaire**
  - **(version simplifiée)**



- **Cycle de vie itératif et incrémental**
  - **Temps: itération**
  - **Produit: incrément**



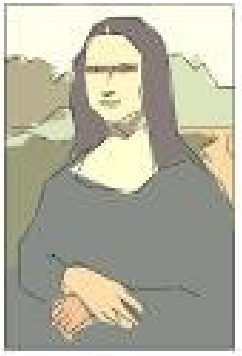
Itératif vs Incrémental

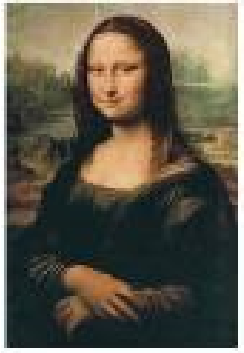
1

2

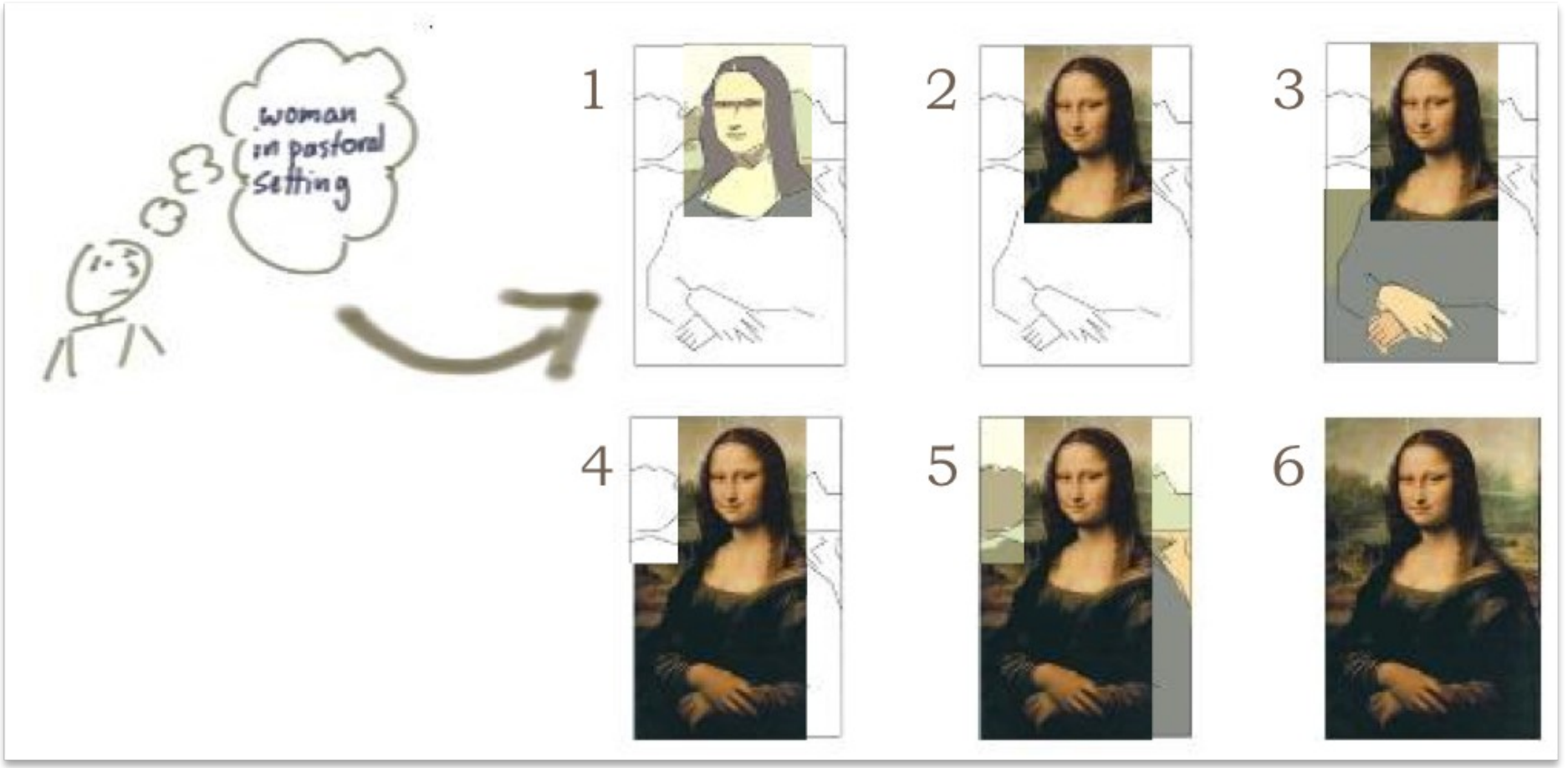
3

1

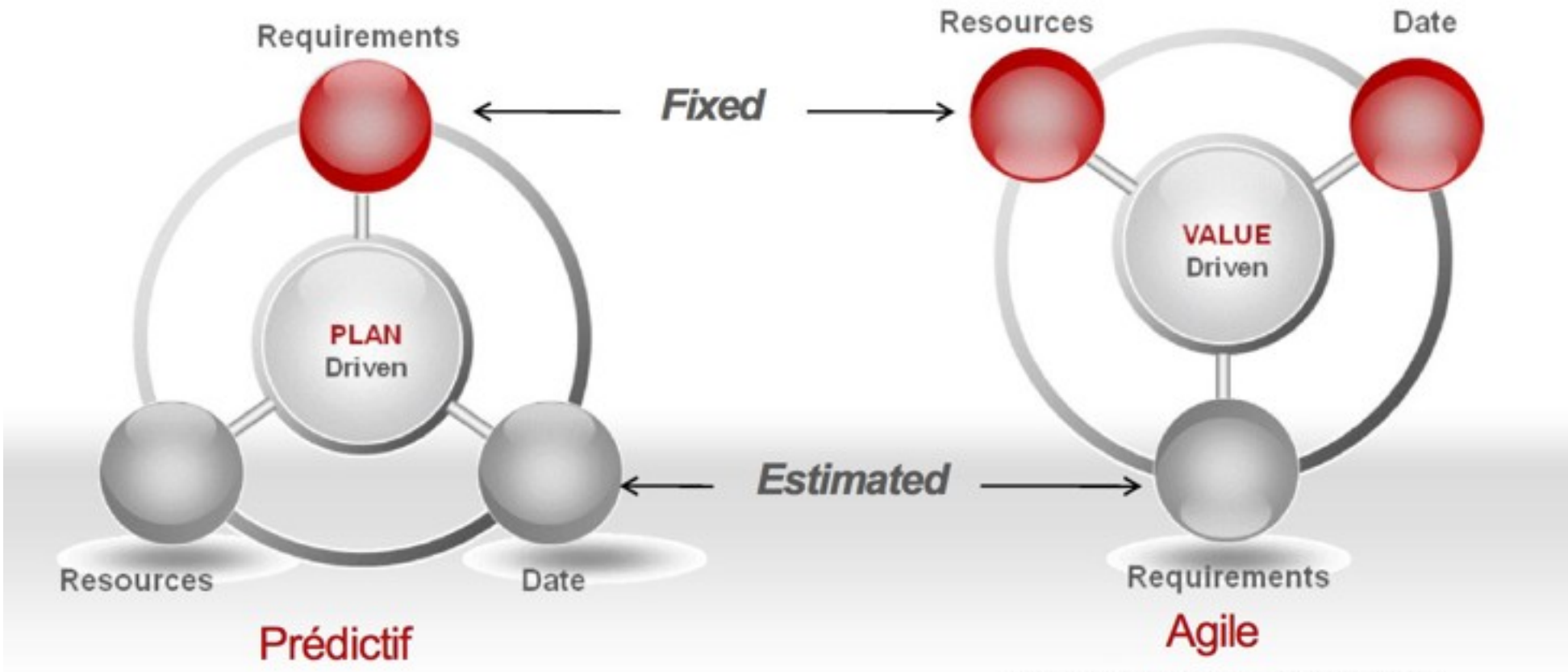
2 

3 

Itératif et Incrémental



Approche classique vs agile



Approche classique vs agile

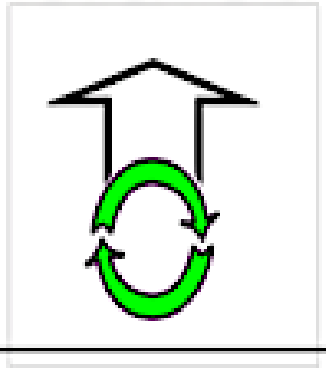
Thème	Approche traditionnelle	Approche agile
Cycle de vie	En cascade ou en V, sans rétroaction possible, phases séquentielles.	Itératif et incrémental.
Planification	Prédictive, caractérisée par des plans plus ou moins détaillés sur la base d'un périmètre et d'exigences définies et stables au début du projet.	Adaptative avec plusieurs niveaux de planification (macro- et microplanification) avec ajustements si nécessaires au fil de l'eau en fonction des changements survenus.
Documentation	Produite en quantité importante comme support de communication, de validation et de contractualisation.	Réduite au strict nécessaire au profit d'incréments fonctionnels opérationnels pour obtenir le feedback du client.

Approche classique vs agile

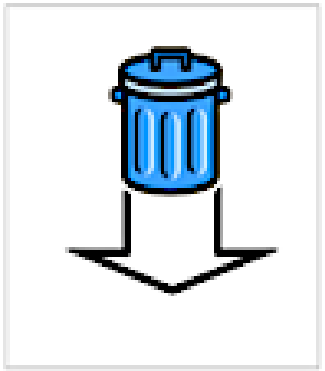
Équipe	Une équipe avec des ressources spécialisées, dirigées par un chef de projet.	Une équipe responsabilisée où l'initiative et la communication sont privilégiées, soutenue par le chef de projet.
Qualité	Contrôle qualité à la fin du cycle de développement. Le client découvre le produit fini.	Un contrôle qualité précoce et permanent, au niveau du produit et du processus. Le client visualise les résultats tôt et fréquemment.
Changement	Résistance voire opposition au changement. Processus lourds de gestion des changements acceptés.	Accueil favorable au changement inéluctable, intégré dans le processus.

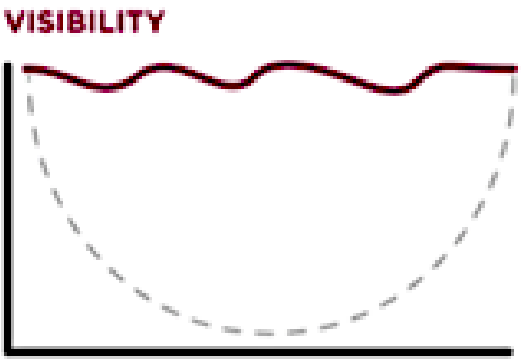
Avantages de l’agile

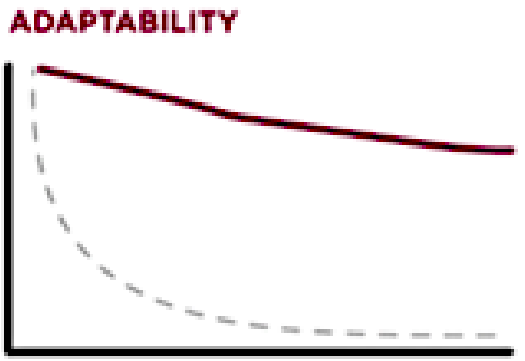
Increase the rate of feedback

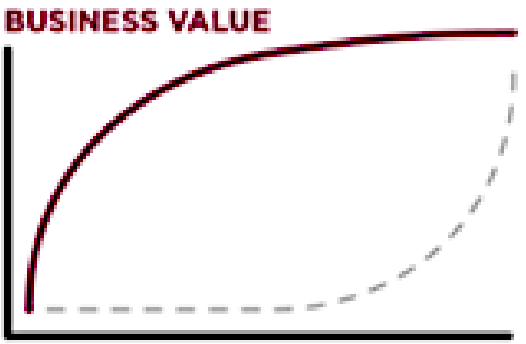


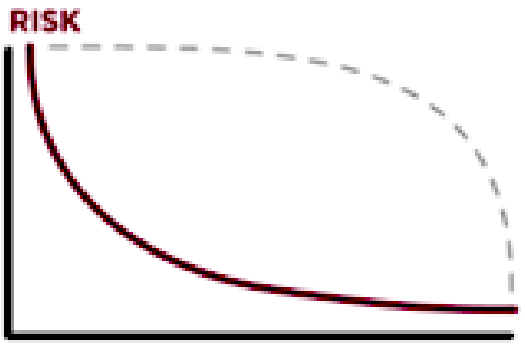
Reduce waste









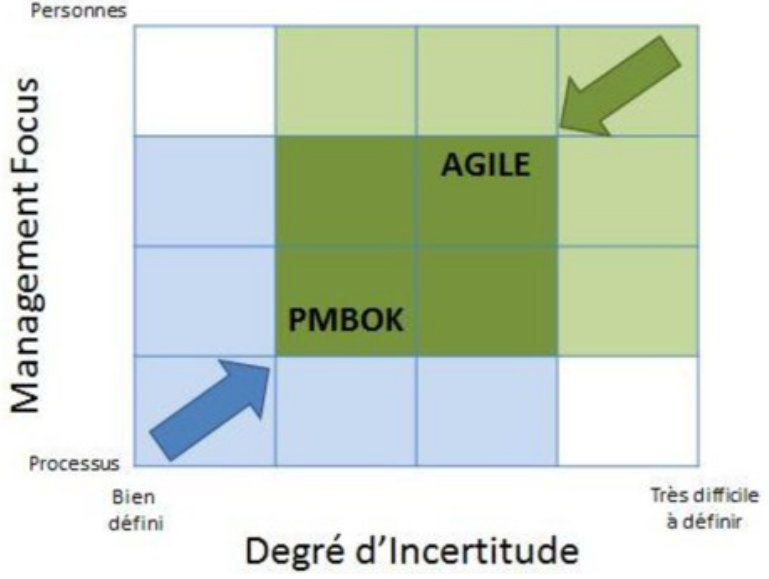


AGILE DEVELOPMENT — — — TRADITIONAL DEVELOPMENT

SCRUM

- 1986 : nouvelle approche qui augmenterait la vitesse et la flexibilité dans le développement de nouveaux produits.
- Le terme Scrum est introduit et provient du rugby (Mêlée)
- 2001 : Méthode décrite dans le livre « Agile Software Development With Scrum»

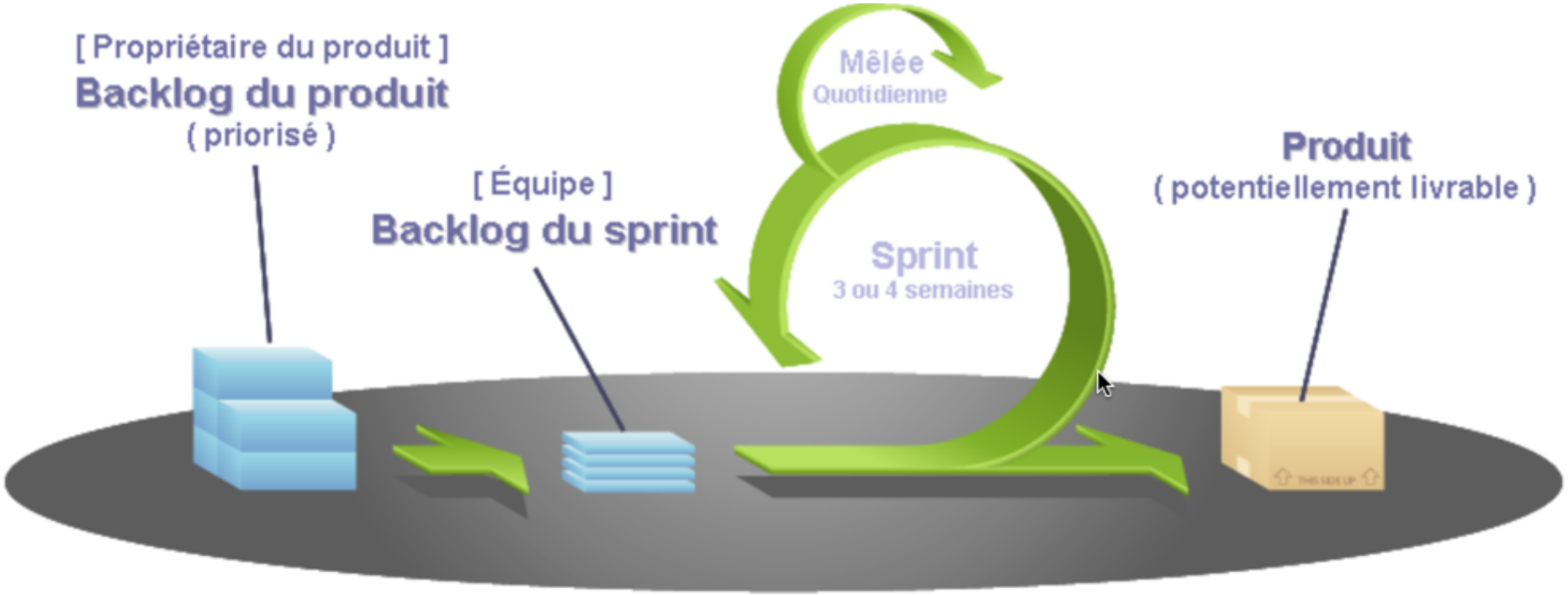
Positionnement des méthodes	Niveau de management
Lean	Business management
Scrum	Project management
Extreme programming	Software engineering



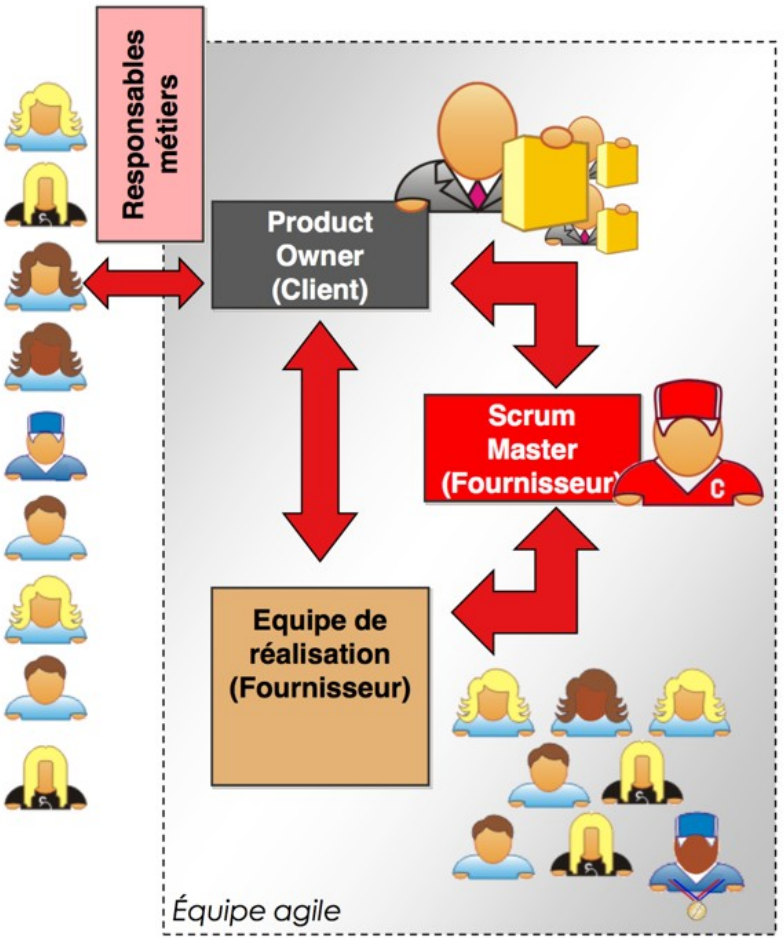
SCRUM

- **Des principes**
  - Série de Sprint ou l'on conçoit, développe et teste la solution
  - Fonctionnement empirique
  - Equipe auto-organisé
- **Des rôles**
  - Product Owner,
  - Scrum Master,
  - Development Team
- **Des cérémonies**
  - Daily Scrum
  - Sprint planning
  - Sprint review
  - Retrospective
- **Des artifacts**
  - Products backlog
  - Sprint backlog
  - Burndown charts

SCRUM


[Propriétaire du produit]
Backlog du produit
(priorisé)
[Équipe]
Backlog du sprint
Mêlée
Quotidienne
Sprint
3 ou 4 semaines
Produit
(potentiellement livrable)

Les rôles



- Product Owner
  - Définit le besoin
  - Priorise le besoin
  - Valide les livraisons
  - N’est pas un supérieur !

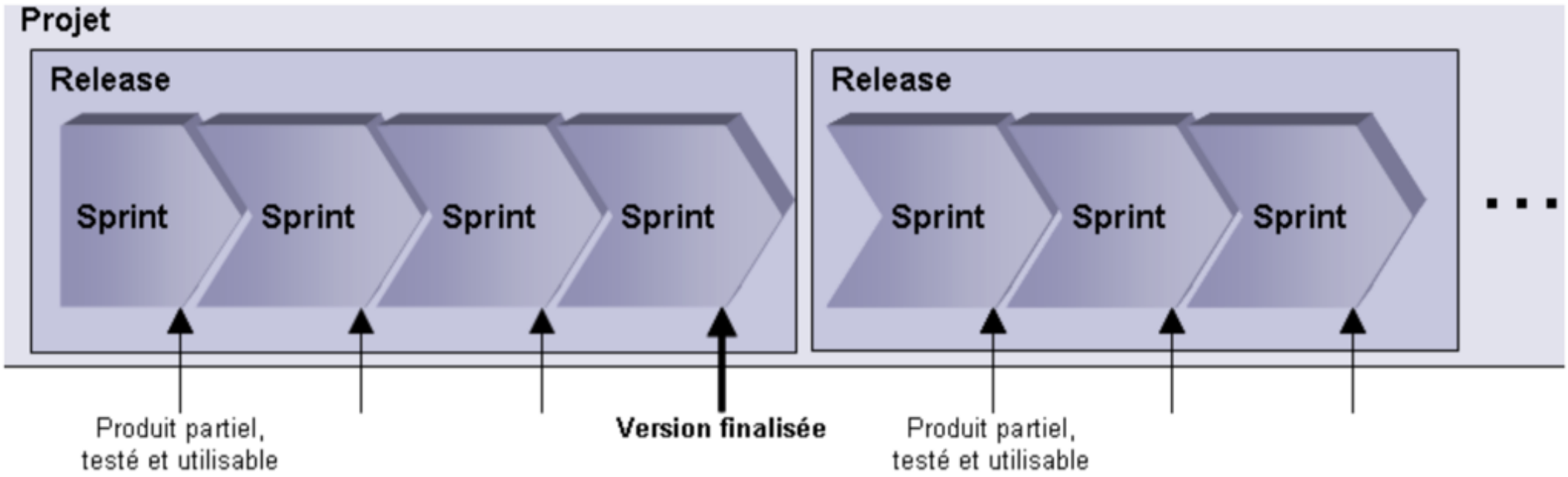
- Scrum Master
  - Garant de la méthode
  - Protège et soutient la team
  - N’est pas un chef de projet !

- Team
  - 5 à 10 pers.
  - Analyse et estime
  - Réalise
  - Auto-organisée
  - Fixe ses propres règles

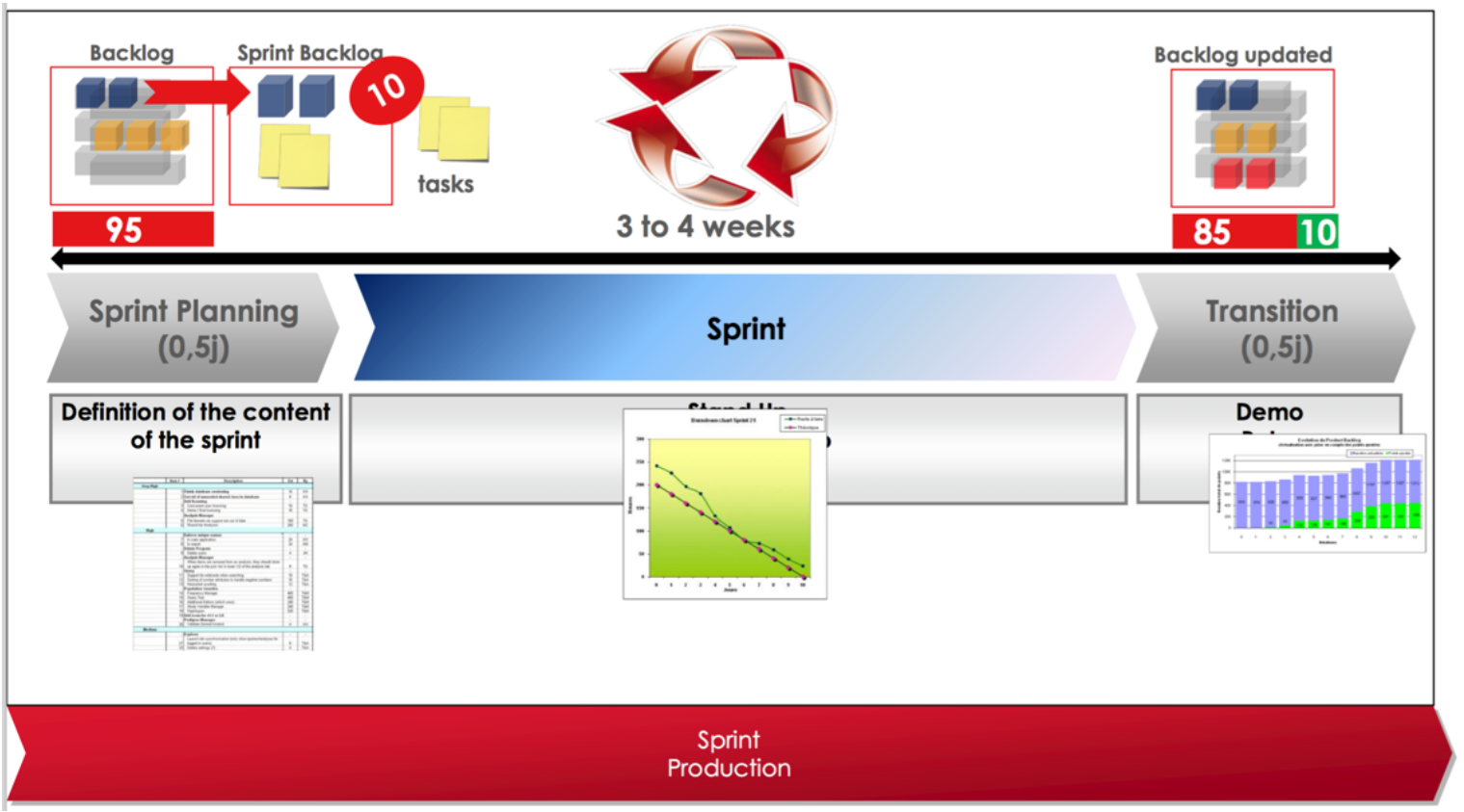
Principes

- Processus simple (4 « cérémonies »)
- Focalisé sur l’organisation
- Sprint : 2 à 4 semaines (fixe)
- Produit fonctionnel (partiellement) à chaque fin de sprint
- Avancement quotidien
- Participation active du client nécessaire

Planification



Planification



Product Backlog

- Ensemble des besoins métier priorisés
- Besoins exprimés en Release/Feature/Epic/User Story
- Fait au début du projet
- Évolue en permanence
- Responsabilité du Product Owner
  - Séances de Story Mapping (avec le métier)

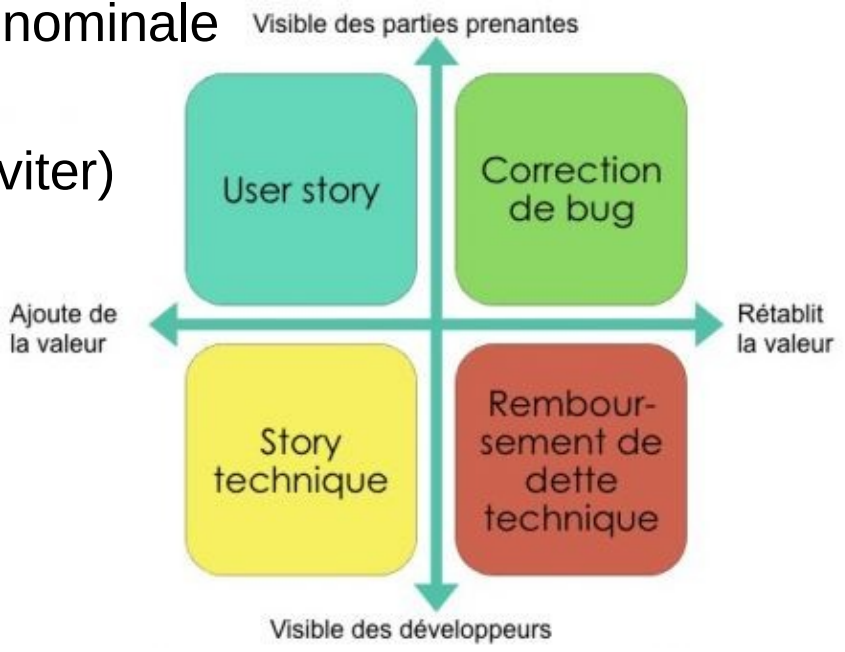
Product Backlog

Project: Shopping Website

Priority	Product Backlog Items	User Story #	User Story	Story Point	Estimate (Hours)
1	Database Creation	9	As an operations engineer, I want to be able store all customer information, so that I can serve to customers.	40	240
2	Login Page	15	As a site member, I want to login the site, so that I can do online shopping.	20	160
3	Category Page	23	As a site member, I want to be able to look for different categories of brands, so that I can choose what I want.	100	400
4	Payment Process	18	As a site member, I will be able to make payment, so that my deliveries can be shipped.	40	240
5	Contact Page	3	As a site member, I want to be able to find contact information of the site, so that in case I need, I can contact.	13	80
6	Banner Area	1	As a marketing personnel, I want to be able to make advertisement, so that I can attract visitors	8	40

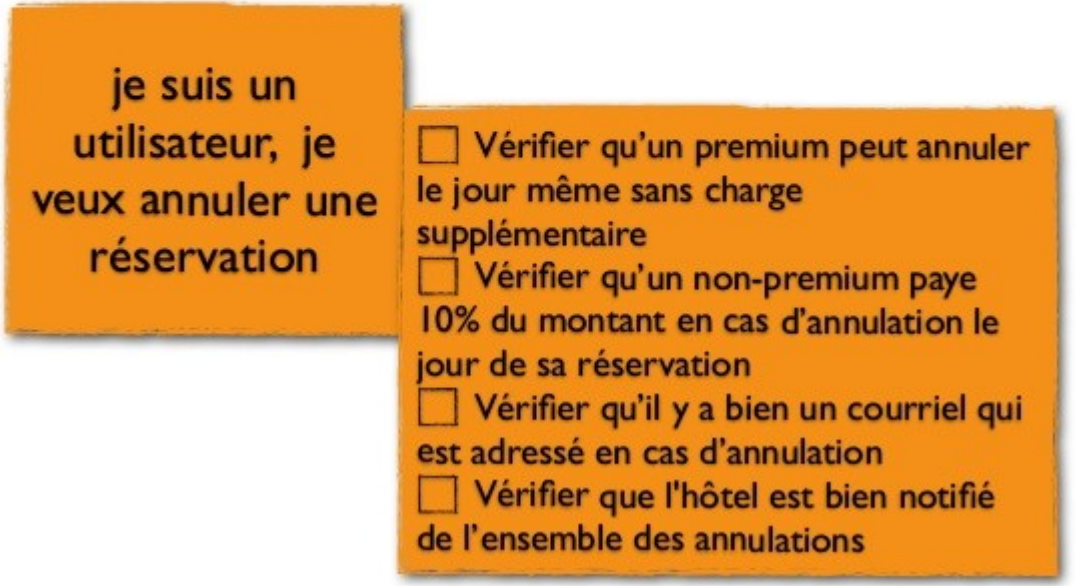
User Story

- Unité de base de Scrum
- Courte phrase décrivant l'ajout de valeur pour le client
- Rédigée par le Product Owner
- La Story est généralement fonctionnelle et nominale
- La Story peut décrire un bug : Defect Story
- Elle peut être plus rarement technique (à éviter)
- Suit généralement le pattern :
  - En tant que <rôle>
  - Je veux/peux <action>
  - Afin de <but>



Les 3 « C »

- Card : la phrase est courte
- Conversation : les détails sont discutés oralement par l’ équipe et le métier
- Confirmation : l’histoire est confirmée par les tests (rédigés si possible sur la carte)



Priorisation

- Gain financier
- Coût de développement
- Satisfaction utilisateurs
- Le Product Owner doit ponderer chaque Story d’une priorité
  - Système MoSCoW
    - MUST
    - SHOULD
    - COULD
    - WOULD

Estimation (Story Points)

- Méthode du Planning Poker
- Faite par l’équipe (au complet)
- Tailles de tee-shirt : XS, S, M, L, XL
- Fibonacci : 1, 2, 3, 5, 8, 13, 21, etc..
- Vote secret
- Les 2 estimateurs extrêmes expliquent leurs choix
- On recommence jusqu’à convergence
- Mieux vaut être vaguement juste que précisément faux !
- Ré-estimation au cas par cas possible en fin de sprint

Propriétés d’une story

Des autres stories

Courte (2ème « C »)

Source de valeur

Par l’équipe

Contient dans une itération

3ème « C »

Epic

- User Story trop large pour entrer dans un Sprint
- Doit être décomposée en plusieurs stories plus petites

Definition Of Done

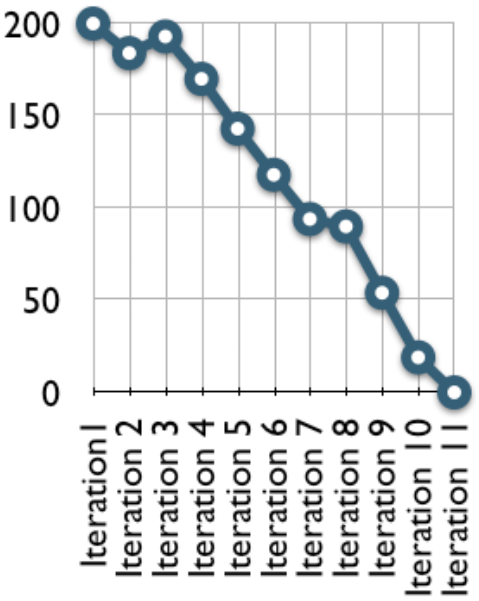
- A définir au début du projet
- Fait par l’équipe
  - Tâche :
    - Code OK et committé
    - Respecte les conventions de nommage
    - Tests unitaires OK
    - Doc à jour
  - Story :
    - Toutes taches sont OK
    - Tests utilisateurs OK
    - Docs à jour

Sprint 0

- Mise en place de l’environnement
- Teste la méthode
- Environ 1 à 2 semaines
- Etalonne la vélocité initiale (SP/h)
- Permet le Release Plan Initial

Release Plan

- Quand toutes les stories sont estimées
- Quand la vélocité initiale est déterminée
- Les itérations/releases sont planifiées (en dates)
- Le Release Burndown Chart est créé

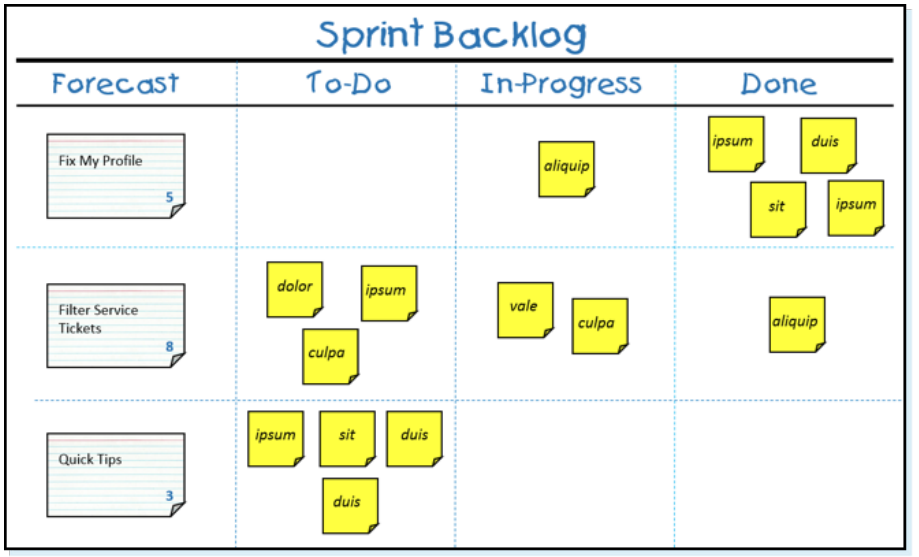


Sprint planning

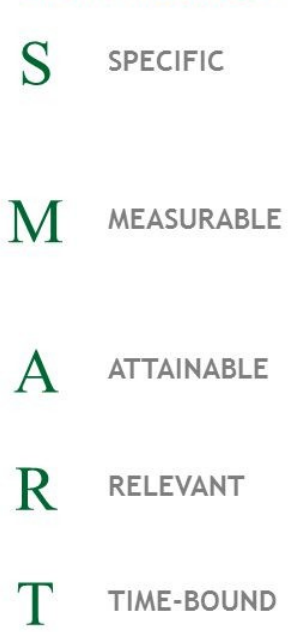
- Pré-requis :
  - DoD OK
  - Story Points OK
  - Priorisation OK
  - Product Backlog OK
  - Tests d’acceptation OK

Sprint planning

- Le PO choisit les stories qu'il désire
- La Team découpe la Story en tâches
- La Team planifie le travail : Qui fait Quoi, ~Gantt, etc
- Le plan est accepté par tous !



Création des tâches



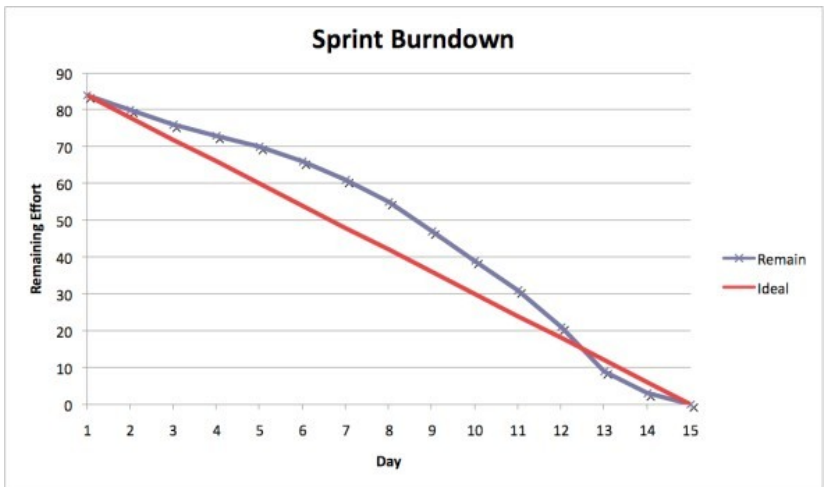
- Simple/Compréhensible
- Mesurable/ « Finissable » -> DoD
- Atteignable (entre dans un sprint – gén. <15h)
- Réaliste – participe à une story
- Chiffrable (compter 30% de distractions)

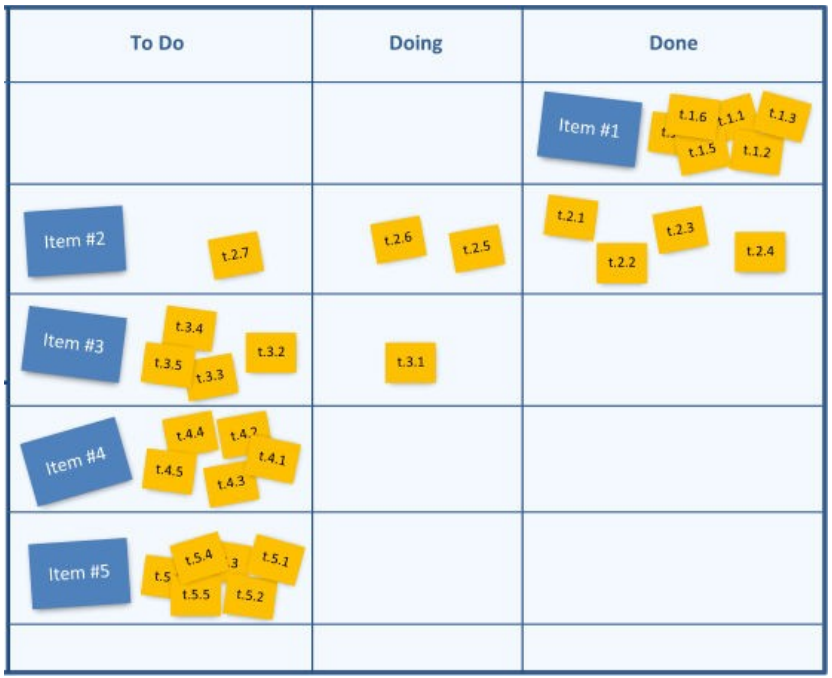
Spikes

- En cas de difficulté à découper une Story ou chiffrer une tâche, la Team crée une Story/tâche technique d’investigation
  - Spike : itération N
  - Story repoussée au sprint N+1
  - Durée limitée (en gén. < 8h)

Sprint Backlog

- Liste de stories d'un sprint
  - Capacité = Vélocité x Durée du sprint
- Décomposition des stories en tâches
- Burndown Chart





Daily Meeting

- Point quotidien à heure fixe (décidé par tous)
  - Team
  - Scrum-Master
- On reste debout : dynamique
- 2 min par personne :
  - Qu’ai-je fait hier ?
  - Que vais-je faire aujourd’hui ?
  - Ai-je des problèmes ?
  - Reste à faire ?
- Aucun hiérarchique autorisé

Daily Meeting

- On reste concentré sur le sprint (stories + tâches)
- Le Scrum Master veille à maîtriser les temps de parole
- Veille au respect du planning
- Mets à jour le Burndown Chart

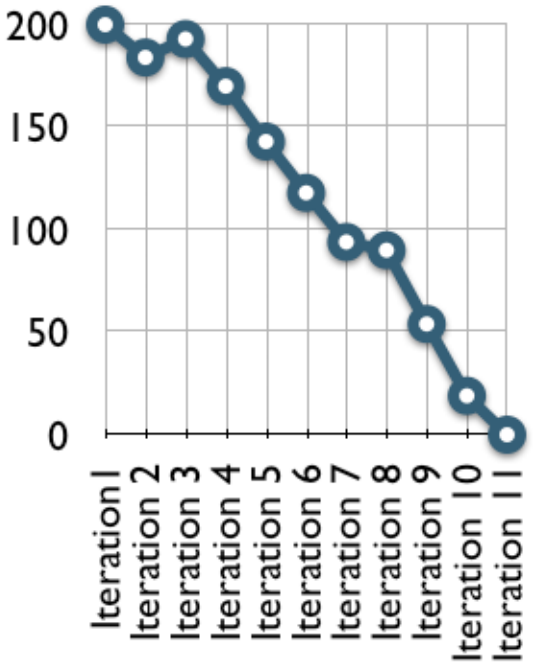
- Avantages :
  - Identification des problèmes au plus tôt
  - Partage des connaissances/avancement
  - Visibilité pour les parties prenantes

Sprint Review

- Bilan du sprint
  - Déroulement du sprint : points forts / difficultés
  - Atteinte des objectifs
- Démonstration du produit livré
  - Présence de quelques utilisateurs conseillée (en plus du PO)

Rétrospective

- Séance de réflexion sur la méthode elle-même
  - Points forts / à garder
  - Axes d'amélioration ?
  - Définir actions correctives (tous ensemble)
  - MàJ du Release Burndown Chart
- Suivie d'un nouveau Sprint Planning !
  - Et on recommence !



Kanban for ITs

- Approche inspirée des méthodes LEAN
- De Scrum à Kanban :
  - L’unité de production n’est plus (forcément) le Sprint mais la Feature/Story
  - Chaque Story est dans un état : Proposée, Acceptée, Mesurable, Estimée, Planifiée, En-Cours, Terminée
  - Chaque état « appartient » à un groupe : PO, Team, Opérationnels, etc..
  - Chaque état à donc une capacité :
    - minimale (chomage technique:))
    - Maximale (production inutile)
    - Un état suivant « tire » une story vers lui (~JIT)
- Déploiement continu (préconisé → DevOps)

eXtreme Programming (XP)

- Axé sur la qualité du code
- Compatible avec Scrum
- Tests Obligatoires ! (TDD min. → BDD mieux !)
- Intégration Continue / Déploiement Continu (CI/CD)
- Amélioration Continue (Code Smell, Refactorings, Dette technique)
- Collaboration (binomes, 4-eyes process)

TDD

- Test Driven Development
- Tests unitaires (Junit,Nunit, xUnit,..)
- A écrire **avant** le code sous test !
- Plateforme d’intégration continue (Jenkins)
- Librairies de mock/stub souvent indispensable !

BDD

- Behavior Driven Development
- Tests d’acceptation automatisés = Specs Auto-testables
- 3 amigos : PO + Tester + Developer
- Librairie : Cucumber, Langage : Gherkin
- Possibité d’exécuter tout type d’application :
  - Desktop
  - Webapp
  - Mobile

```
Feature: Basic Arithmetic

  Background: A Calculator
    Given a calculator I just turned on

  Scenario: Addition
  # Try to change one of the values below to provoke a failure
    When I add 4 and 5
    Then the result is 9

  Scenario: Another Addition
  # Try to change one of the values below to provoke a failure
    When I add 4 and 7
    Then the result is 11

  Scenario Outline: Many additions
    Given the previous entries:
      | first | second | operation |
      | 1     | 1      | +         |
      | 2     | 1      | +         |
    When I press +
    And I add <a> and <b>
    And I press +
    Then the result is <c>

  Examples: Single digits
    | a | b | c  |
    | 1 | 2 | 8  |
    | 2 | 3 | 10 |
```

```
public class RpnCalculatorStepdefs {
    private RpnCalculator calc;

    @Given("a calculator I just turned on")
    public void a_calculator_I_just_turned_on() {
        calc = new RpnCalculator();
    }

    @When("I add {int} and {int}")
    public void adding(int arg1, int arg2) {
        calc.push(arg1);
        calc.push(arg2);
        calc.push("+");
    }

    @Given("I press (.+)")
    public void I_press(String what) {
        calc.push(what);
    }

    @Then("the result is {int}")
    public void the_result_is(double expected) {
        assertEquals(expected, calc.value());
    }
```

CI / CD

- CI
  - Ex : Jenkins
  - Détecte un commit de feature « done »
  - Joue les tests
  - Package et stocke le livrable
- CD
  - Ex : Chef, Puppet, Salt, Ansible,…
  - Détecte un nouveau livrable
  - Le déploie dans l’environnement adéquat : staging, pre-prod, prod...

Rôles

- Programmeur : concepteur et/ou codeur et/ou testeur
- Client : intégré à l’équipe (~PO)
- Testeur : conçoit et joue et automatise les tests
- Coach : garant de la méthode (~SM)
- Tracker : suit l’avancement des taches (~SM)
- Manager : Chef de projet / Hiérarchique

Valeurs

- Courage :
  - Admettre ses erreurs
  - Savoir renoncer/recommencer
- Communication / Feedback
  - Interne
  - externe
- Respect (→ LEAN)

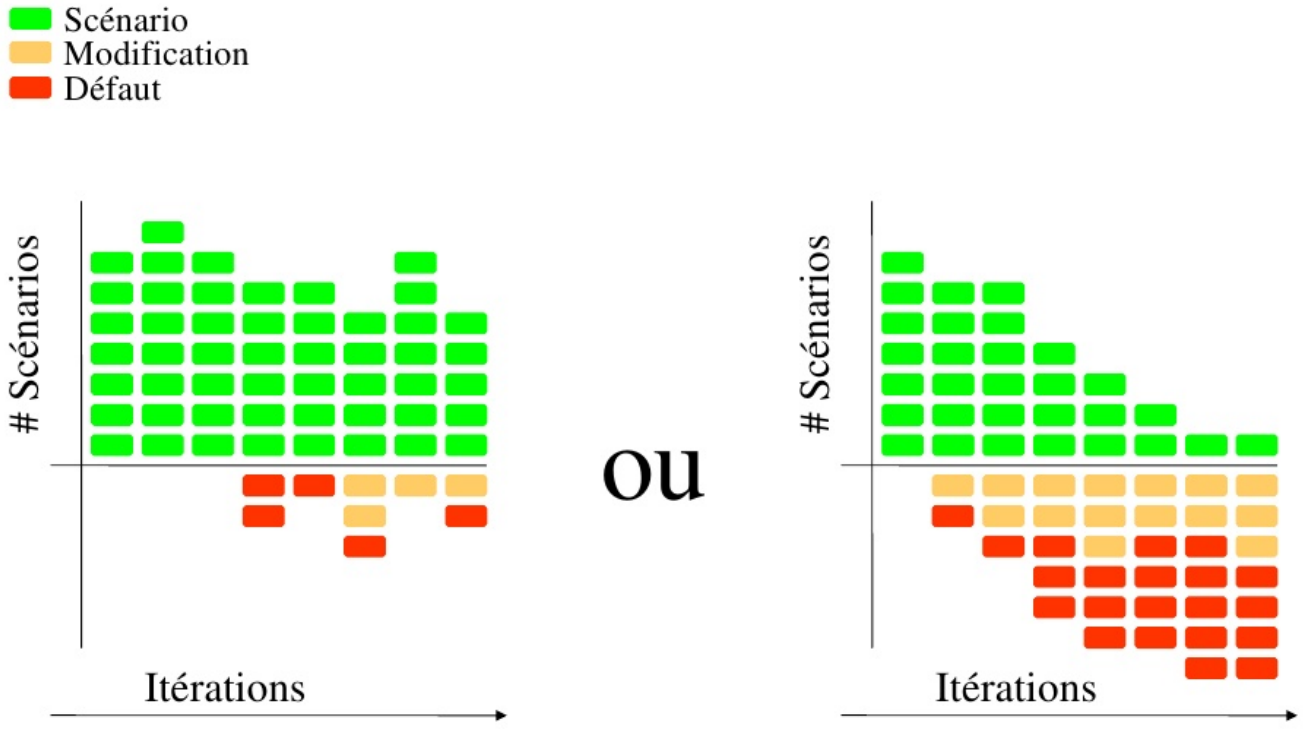
Code Smells

- Conventions de nommage pas respectées
- Trop de paramètres
- Méthodes trop longues
- Classes trop longues
- Types primitifs
- Couplage d’attributs
- If/Switch < Polymorphisme
- Principe DRY pas respecté
- Principe YAGNI pas respecté / Code Mort
- Principe KISS pas respecté
- Principes SOLID pas respectés

Principes SOLID

- Responsabilité unique (Single responsibility)
  - une classe (ou une méthode) doit avoir une et une seule responsabilité
- Ouvert/fermé (Open/closed)
  - une classe doit être ouverte à l'extension, mais fermée à la modification
- Substitution de Liskov (Liskov substitution)
  - une instance de type T doit pouvoir être remplacée par une instance de type G, tel que G sous-type de T, sans que cela ne modifie la cohérence du programme
  - Si ça marche avec une classe mère, ça doit marcher avec une classe fille !
- Ségrégation des interfaces (Interface segregation)
  - préférer plusieurs interfaces spécifiques plutôt qu'une seule interface générale
- Inversion des dépendances (Dependency inversion)
  - il faut dépendre des abstractions, pas des implémentations

Dette Technique



http://www.xprogramming.com/xpmag/BigVisibleCharts.htm