

Agile Scrum (initiation)

Thomas Aldaitz
taldaitz@dawan.fr

Plus d'informations sur <http://www.dawan.fr>
Contactez notre service commercial au **09.72.37.73.73** (appel gratuit depuis un poste fixe)



Objectifs



- Découvrir les méthodes agiles
- Maîtriser les concepts de gestion de projets agiles

Plan



- Agilité
- Cadre Scrum
- Gestion de projet agile
- Besoins et planification
- De l'analyse à l'implémentation
- Outils de gestion

Pourquoi l'agilité ?



Gestion de projets

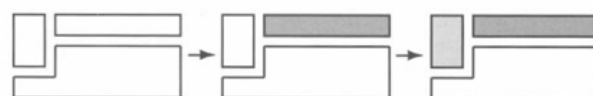
Il existe 2 types d'approches :

- **Approche prédictive** : prévoir des phases séquentielles avec un engagement sur un planning précis de réalisation du projet.
- **Approche agile** : construire un processus itératif et incrémental qui consiste à découper le projet en plusieurs étapes qu'on appelle « itérations ».

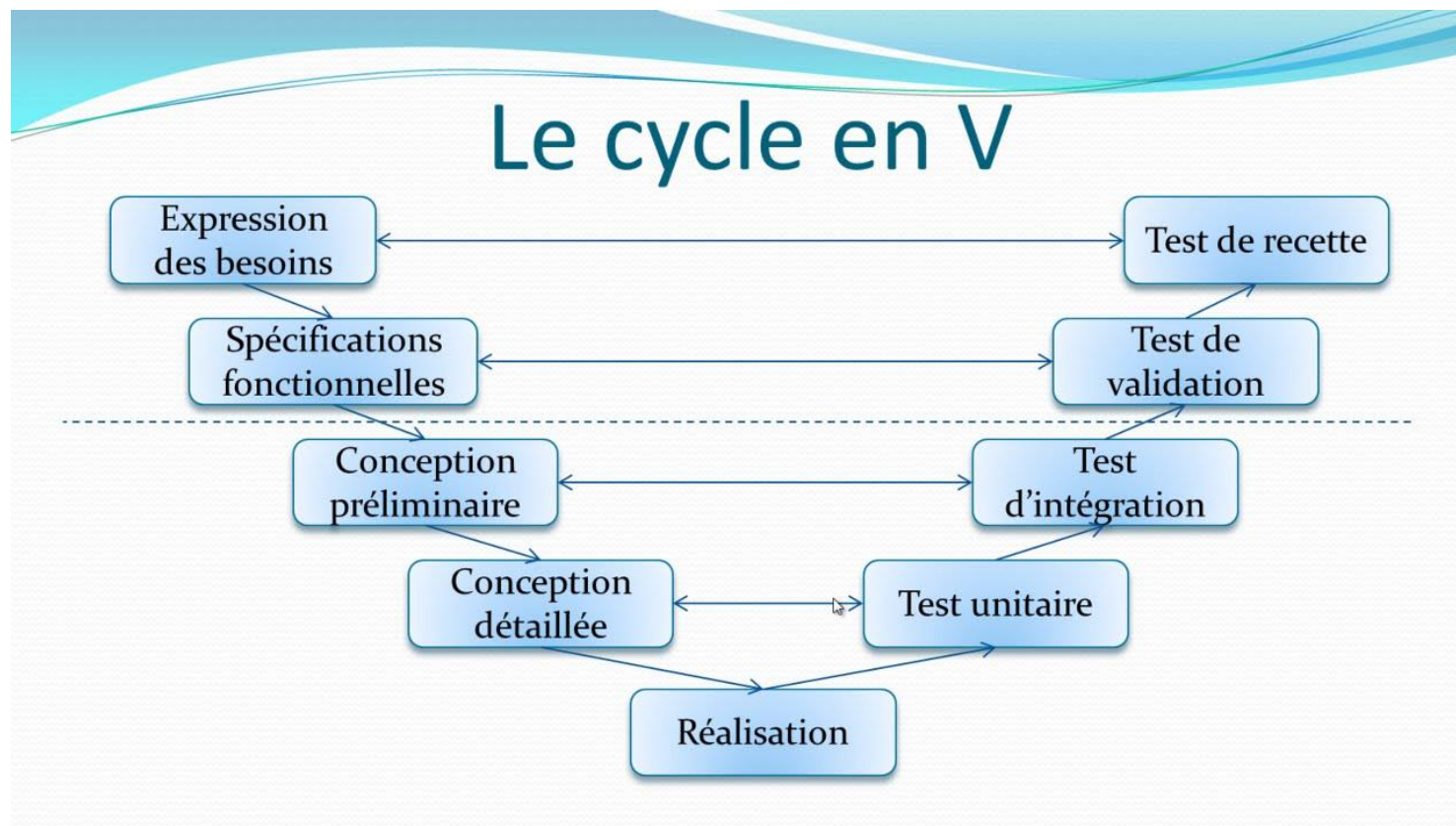
Développement incrémental



Développement itératif



D'où l'on vient



Pourquoi l'agilité ?

- Réduire le temps de mise sur le marché
(Time to market)
- Améliorer la qualité du produit
- Réduire les activités sans réelle valeur ajoutée
- Travailler sur des éléments de plus grande valeur
- Avoir une meilleure prévisibilité
- Améliorer le contexte de travail
(meilleure collaboration / communication)

Critères d'éligibilité

• Favorisants :

- - Besoin rapide de mise à disposition du produit
- - Imprévisibilité des besoins du client
- - Nécessité de changements fréquents
- - Besoin de visibilité du client sur l'avancement des développements
- - Présence de l'utilisateur assurant un feedback immédiat

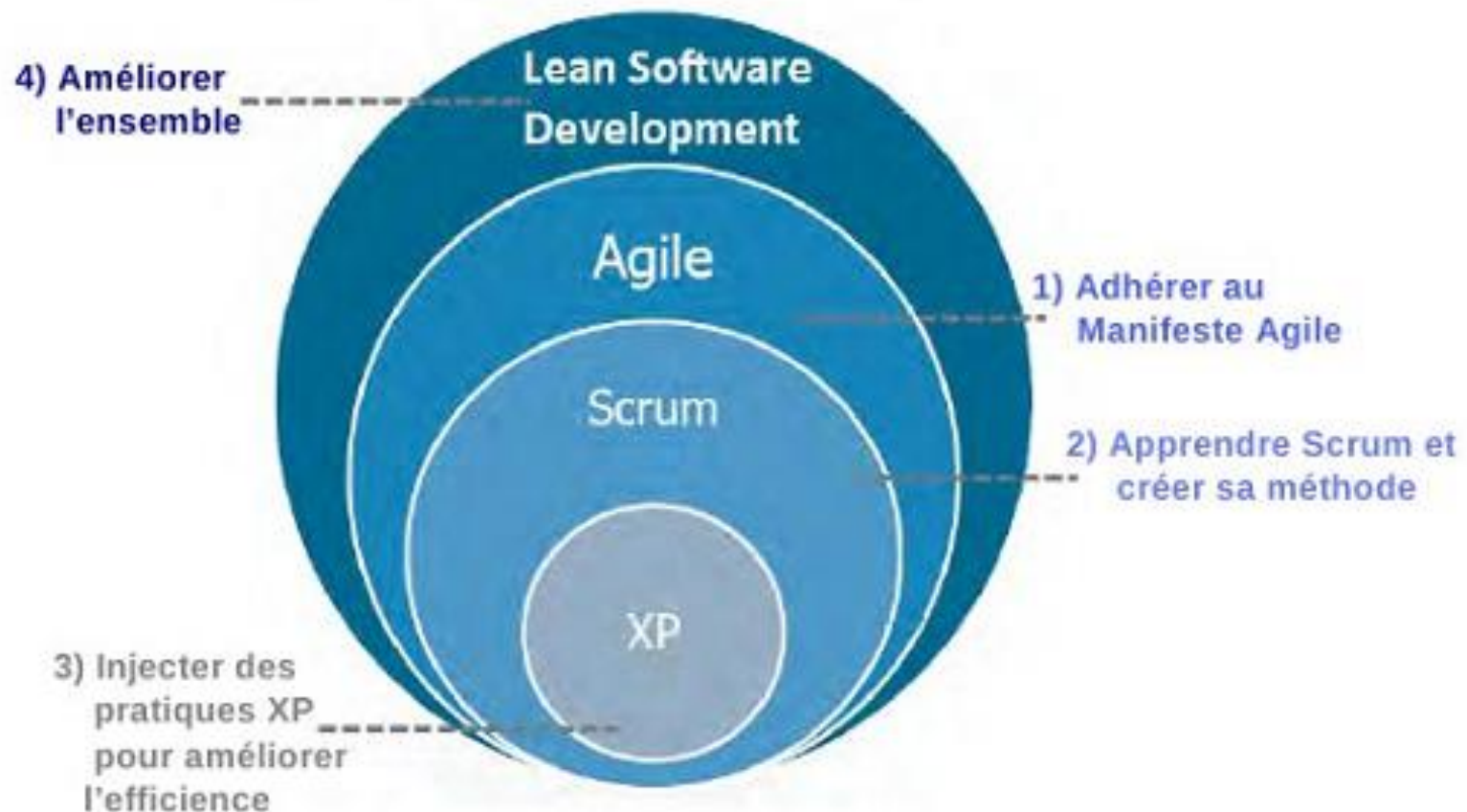
• Défavorisants :

- - Indisponibilité du client ou de l'utilisateur
- - Dispersion géographique des ressources humaines
- - Inertie des acteurs du projet ou refus des changements
- - Gouvernance complexe de la DSI

Origine

- Les suites du *Standish Group Chaos Report* (1994)
- Mouvement né en 2001 aux USA.
- Réunion de 17 experts en développement logiciel.
- But : trouver un socle commun de valeurs et de bonnes pratiques.
- Résultats : écriture du **Manifeste pour le développement logiciel Agile** :
<http://agilemanifesto.org/iso/fr/manifesto.html>
- et création de l'**Agile Alliance** (association chargée de la promotion de l'agilité et du soutien aux équipes) :
<http://www.agilealliance.org/>

Démarche d'adoption



Manifeste Agile

« Nous découvrons de meilleures approches pour le développement informatique en en faisant nous-mêmes, ainsi qu'en aidant les autres dans leur travail ».

Les **valeurs** mises en avant à gauche n'excluent pas les valeurs à droite :

- les individus et leurs interactions avant les processus et les outils.
- des fonctionnalités opérationnelles avant la documentation.
- collaboration avec le client plutôt que contractualisation des relations.
- acceptation du changement plutôt que conformité au plan.

Manifeste Agile

« 12 Principes »



Notre priorité est de satisfaire le client en lui livrant très tôt et régulièrement des versions opérationnelles de l'application, source de valeur.

- Développement itératif, livraison intermédiaire.
- Validation par le client et feedback.
- Conformité de la livraison aux attentes ou prise en compte des remarques et re-priorisation.

Manifeste Agile

« 12 Principes » (2)



Accepter le changement dans les exigences, même tard dans le cycle de vie, pour garantir la compétitivité du client.

- État d'esprit de l'équipe Agile.
- Capacité à comprendre et apprendre comment satisfaire encore mieux la demande.

Manifeste Agile

« 12 Principes » (3)



Livrer le plus souvent possible des versions opérationnelles de l'application, à une fréquence allant de 2 semaines à 2 mois.

- 1 version intermédiaire du produit final, visible et testable satisfait davantage le client qu'une documentation à valider.
- Il a la preuve tangible que le projet avance.
- Il peut exprimer son point de vue sur le résultat présenté

Manifeste Agile

« 12 Principes » (4)



Client et développeurs doivent coopérer quotidiennement tout au long du projet.

- Les relations conflictuelles ne font pas partie de l'esprit Agile.
- Les relations collaboratives et partenariales basées sur la confiance et le consensus sont préférables.
- Le client (ou son représentant) est accessible et disponible, totalement impliqué dans toutes les phases du projet.

Manifeste Agile

« 12 Principes » (5)



**Construire des projets autour d'individus motivés.
Leur donner l'environnement et le support dont ils ont
besoin et leur faire confiance pour remplir leur
mission.**

- Le facteur clé du succès d'un projet est l'équipe.
- Tout obstacle à son bon fonctionnement devra être levé.
- Un changement, s'il s'avère nécessaire, sera apporté aux processus, aux outils, à l'environnement, à la composition de l'équipe.

Manifeste Agile

« 12 Principes » (6)



La méthode la plus efficace de communiquer des informations à une équipe et au sein de celle-ci reste la conversation en face à face.

- Par défaut, on privilégie l'oral à l'écrit, pour lever toute ambiguïté et favoriser la rapidité de la compréhension.
- Tout ne peut être formalisé par écrit, notamment la “connaissance tacite”, c'est à dire, “l'information informelle”, la culture du projet, détenues par chacun au sein d'une équipe.

Manifeste Agile

« 12 Principes » (7)



Le fonctionnement de l'application est le premier indicateur d'avancement du projet.

- Il n'existe pas d'autre indicateur plus pertinent que le pourcentage ou le nombre d'exigences satisfaites.
- On ne mesure pas un projet à la quantité de documents produits ou au nombre de lignes de code, non significatifs pour le client.

Manifeste Agile

« 12 Principes » (8)



Les méthodes agiles recommandent que le projet avance à un rythme soutenable.

- La qualité du travail fourni dépend du rythme du travail qui doit être adapté en fonction des spécificités du projet
- Le rythme doit être soutenu et soutenable sur la durée du projet.

Manifeste Agile

« 12 Principes » (9)



Sponsors, développeurs et utilisateurs devraient pouvoir maintenir un rythme constant indéfiniment.

- Ce rythme de travail est à déterminer par l'ensemble des membres de l'équipe et par le client, en fonction de la productivité de l'équipe et des priorités du client.
- Travailler en heures supplémentaires pour corriger des bogues ou des régressions n'apporte aucune valeur ajoutée.

Manifeste Agile

« 12 Principes » (10)



Porter une attention continue à l'excellence technique et à la conception améliore l'agilité.

- Maintenir un code propre, évolutif et performant est un objectif permanent de l'équipe.
- Il ne s'agit pas de produire du code non exploitable par les autres, ni du jetable.
- Cela évite d'enliser les développements ultérieurs, avec des modifications cassant un développement fragile, nécessitant des interventions à des endroits variés du code.

Manifeste Agile

« 12 Principes » (11)



La simplicité (art de maximiser la quantité de travail non fait) est essentielle.

- La simplicité garantit l'évolutivité du système.
- La complexité, au contraire coûte davantage et rend plus difficiles les évolutions inhérentes au développement incrémental.
- La conception doit ne comporter que des éléments utiles.

Manifeste Agile

« 12 Principes » (12)



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

- Le chef de projet n'est pas celui qui attribue des tâches.
- L'équipe, elle-même, se responsabilise et définit ses travaux à réaliser.
- Le partage des tâches s'effectue sur la base du volontariat.

Méthode agile

- Approche itérative et incrémentale
- Esprit collaboratif
- Formalisme léger
- Produit de haute qualité
- Acceptation du changement

Approche itérative et incrémentale



- Découper le projet en plusieurs étapes d'une durée de quelques semaines : **les itérations (sprints)**
- Au cours d'une itération, une version minimale du produit est développée puis soumise au client pour validation.
- Les fonctionnalités sont ainsi intégrées au fur et à mesure du cycle de vie sur un mode incrémental.
- Le système s'enrichissant progressivement pour atteindre les niveaux de satisfaction et de qualité requis.
- **Chaque itération est un mini-projet** : activités de développement menées en parallèle – analyse, conception, codage et test - activités de management de projet.

- Le résultat d'une itération n'est pas un prototype ou une « proof of concept » mais bien une version intermédiaire du produit final.
- Les itérations se succèdent et ne peuvent être parallélisées :
 - - Avancer prudemment et s'adapter au fur et à mesure
 - - Pas de plan de management de projet unique mais une liste de besoins macroscopiques et macroplanning initial (grandes échéances et jalons principaux).

- Principe du timeboxing (boîte de temps ou tranche de temps) :
 - * Date d'échéance fixe immuable pour l'itération.
 - * Mobilisation des efforts sur des objectifs clairs à court terme.
 - * Si les objectifs ne sont pas atteints, les enseignements seront tirés lors du bilan de l'itération pour corriger les conditions de l'itération suivante.

Avantages :

- La communication est de meilleure qualité.
- La visibilité est meilleure.
- La qualité est évaluée en continu.
- Les risques sont détectés très tôt.
- L'équipe prend confiance.
- Les coûts sont contrôlés.

Méthode agile

Esprit Collaboratif



- Placer les individus et leurs interactions au centre du dispositif.
- Communication entre les différents acteurs d'un projet, au sein de l'équipe, mais aussi entre l'équipe et le client / les utilisateurs.
- Respect des opinions des autres.
- Capacité à exprimer des opinions différentes de façon non agressive.
- Aptitude à rechercher et atteindre les consensus sans frustration.
- Prédilection à l'autodiscipline, voire à l'autogestion.
- Créativité et performance de l'équipe.

Méthode agile

Formalisme léger



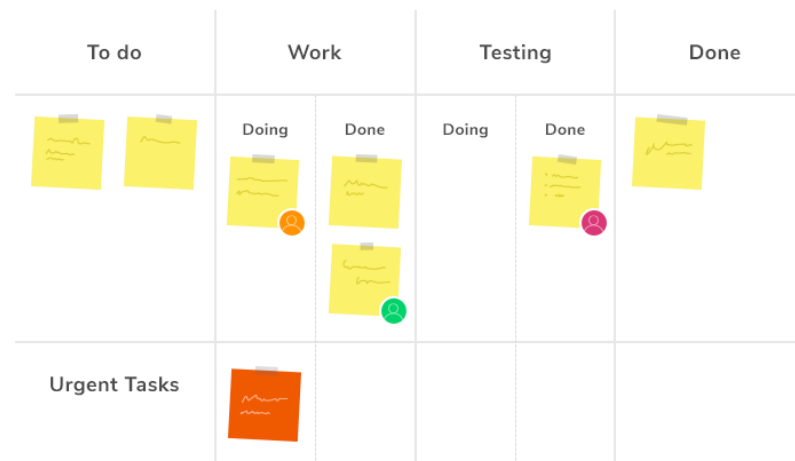
- L'essentiel: les versions intermédiaires du produit.
- Quelques livrables à produire.
- Quelques rôles définis.
- Quelques étapes
- Quelques réunions.
- Quelques outils simples.
- ... et la démarche est formalisée.

Pratiques agiles



Kanban

- **Système visuel de gestion des processus qui indique quoi produire, quand le produire et en quelle quantité**
- Elle emploie un système de tirage limité de tâches en cours comme mécanisme central, afin de déterminer les processus du système et stimuler la collaboration dans le but d'une amélioration continue du système.



Kanban (principes)



- **Commencer par ce que vous faites actuellement**

Commencer avec les rôles et processus déjà définis et stimule des changements continus, augmentés et évolutifs.

- **Accepter d'appliquer les changements évolutifs et augmentés**

L'équipe doit accepter que les changements continus, augmentés et évolutifs sont le moyen d'améliorer le système.

- **Respecter le processus actuel, les rôles et les responsabilités**

Les changements futurs doivent être facilités et le respect des rôles, des responsabilités et des titres professionnels actuels permettent d'éliminer les peurs initiales.

- **Leadership à tous les niveaux**

Les actes de leadership à tous les niveaux au sein de l'organisation, qu'il s'agisse de collaborateurs indépendants ou de cadres supérieurs doivent être encouragés.

Kanban (pratiques)



•Visualiser

La visualisation du workflow et sa matérialisation permettent de comprendre comment fonctionnent les processus. Un moyen courant de visualiser le workflow est d'utiliser un tableau avec des colonnes (Kanban Tasks Board).

•Limiter le nombre de tâches en cours

Un système de tirage est mis en application sur le workflow. Ce système servira de stimulus principal pour les changements continus.

•Gestion du flux

Le déroulement du travail à travers chaque stade du workflow doit être suivi, mesuré et rapporté.

•Rendre les normes de processus explicites

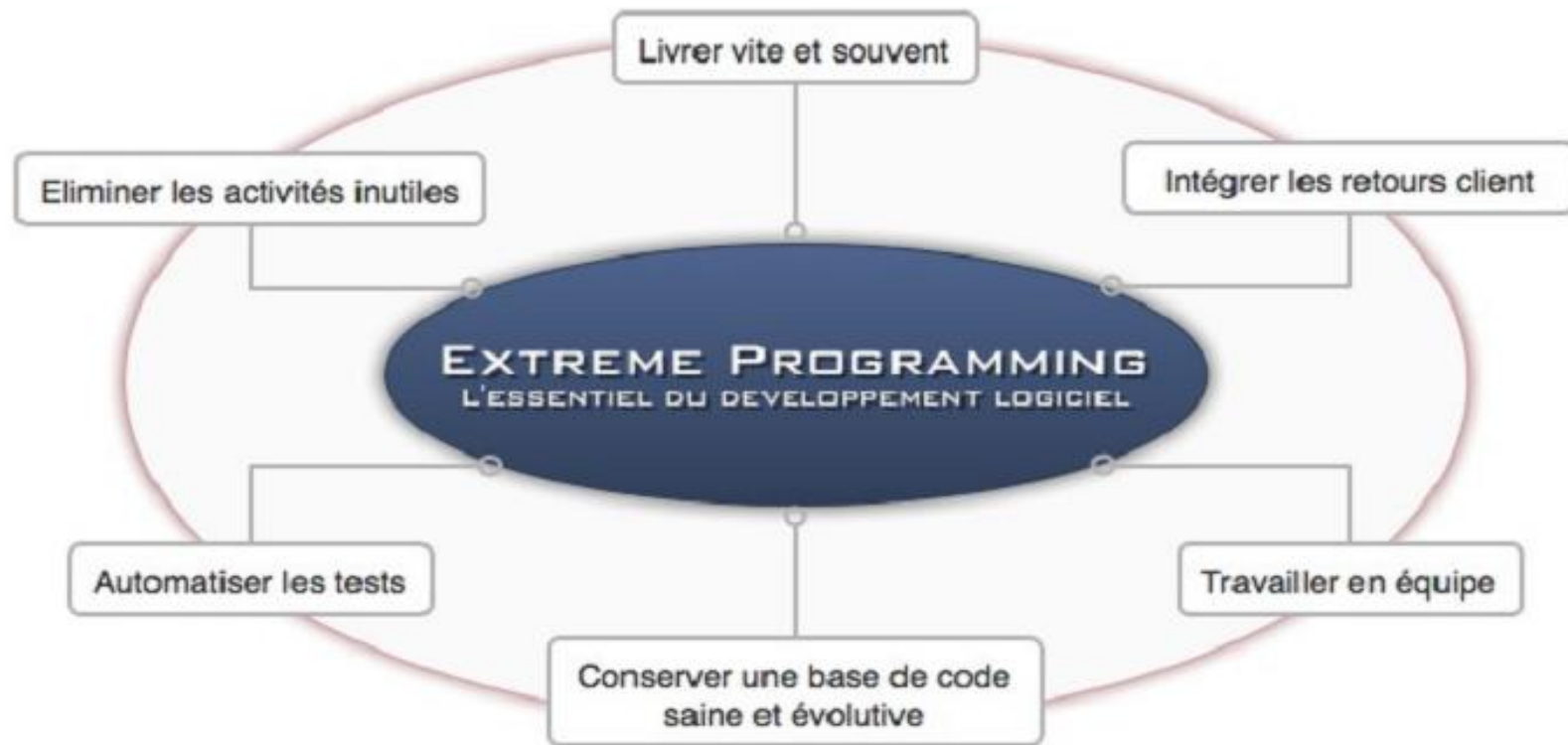
Établir les règles et recommandations par la compréhension des besoins et le suivi des règles (quand et pourquoi un ticket est déplacé).

•Utiliser des modèles pour reconnaître les opportunités d'amélioration

Lorsque les équipes partagent une compréhension des théories sur le workflow, le processus et le risque, elles pourront comprendre les problèmes et proposer des actions d'amélioration.

XP (eXtreme Programming)

Créé en 1996 par Kent BECK et Ron JEFFRIES



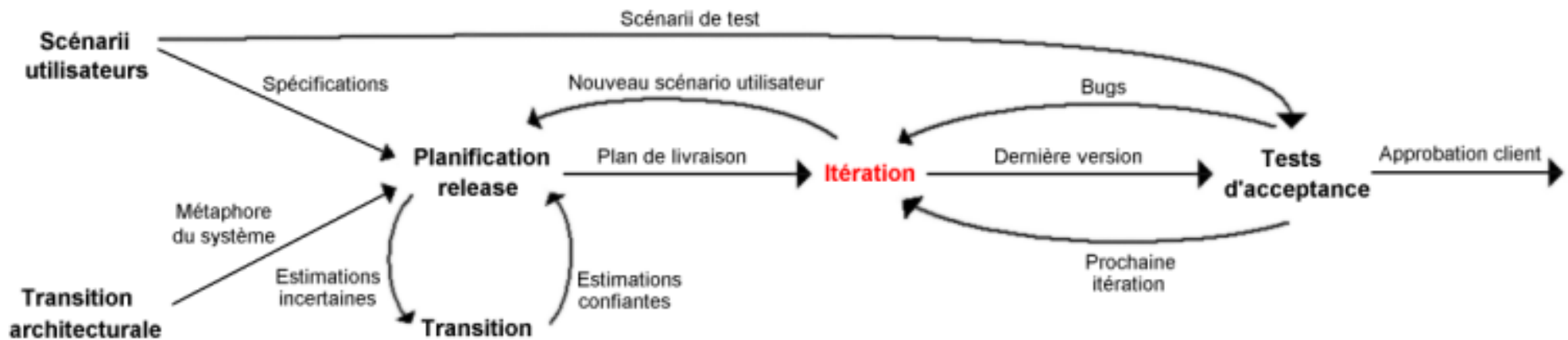
XP (eXtreme Programming)

Principes => Pratiques



- Planning :
 - phase d'exploration : écriture des besoins (user stories) et estimation.
 - phase d'engagement : classement des user stories par ordre de priorité.
 - phase de direction : mise à jour du planning.
- Petites releases
- Utilisation de métaphores pour décrire l'architecture du système.
- Conception simple : toujours développer la solution la plus simple possible.
- Tests (unitaires et fonctionnels)
- Refactoring du code: retravailler le code pour plus de lisibilité et de robustesse.
- Programmation en binôme.
- Propriété collective du code.
- Intégration continue (plusieurs fois par jour).
- Pas de surcharge de travail : ne pas dépasser 40h de travail par semaine.
- Client sur site : présence sur site d'une personne minimum à temps plein pendant toute la durée du projet.
- Standards de code (normes de nommage et de programmation).

XP (eXtreme Programming) Cycle de vie



XP (eXtreme Programming)

Rôles



- Développeur
- Client
- Testeur
- Tracker
- Coach
- Manager

Lean

(Lean software management)



Amélioration continue, dans le respect des collaborateurs et partenaires, pour la recherche de l'excellence :

- Challenger sa performance tous les jours.
- Voir les problèmes.
- Résoudre les problèmes.
- En tirer les bons enseignements.

Origine : Toyota

Lean

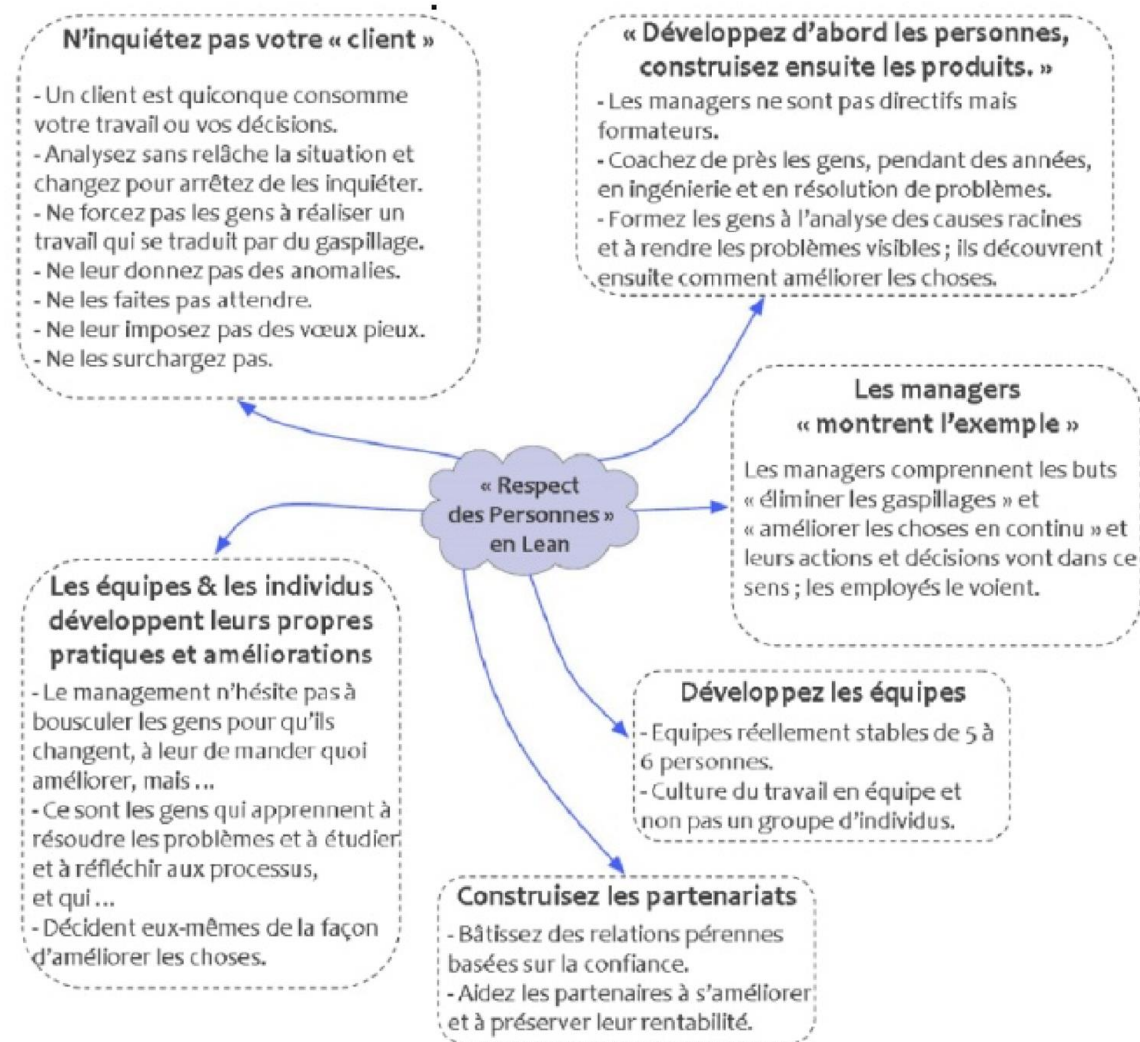
« Principes »



- **Optimiser le système dans son ensemble**
 - - Approche à long terme.
- **Travailler en mode flux**
 - - Cycle court
 - - Livrer rapidement
 - - Visibilité régulière (Management Visuel)
- **Décider le plus tard possible**
 - - Reporter la décision "Last Responsible Moment"
- **Lisser le travail**
- **Maîtriser les standards**

Lean

Respect des personnes



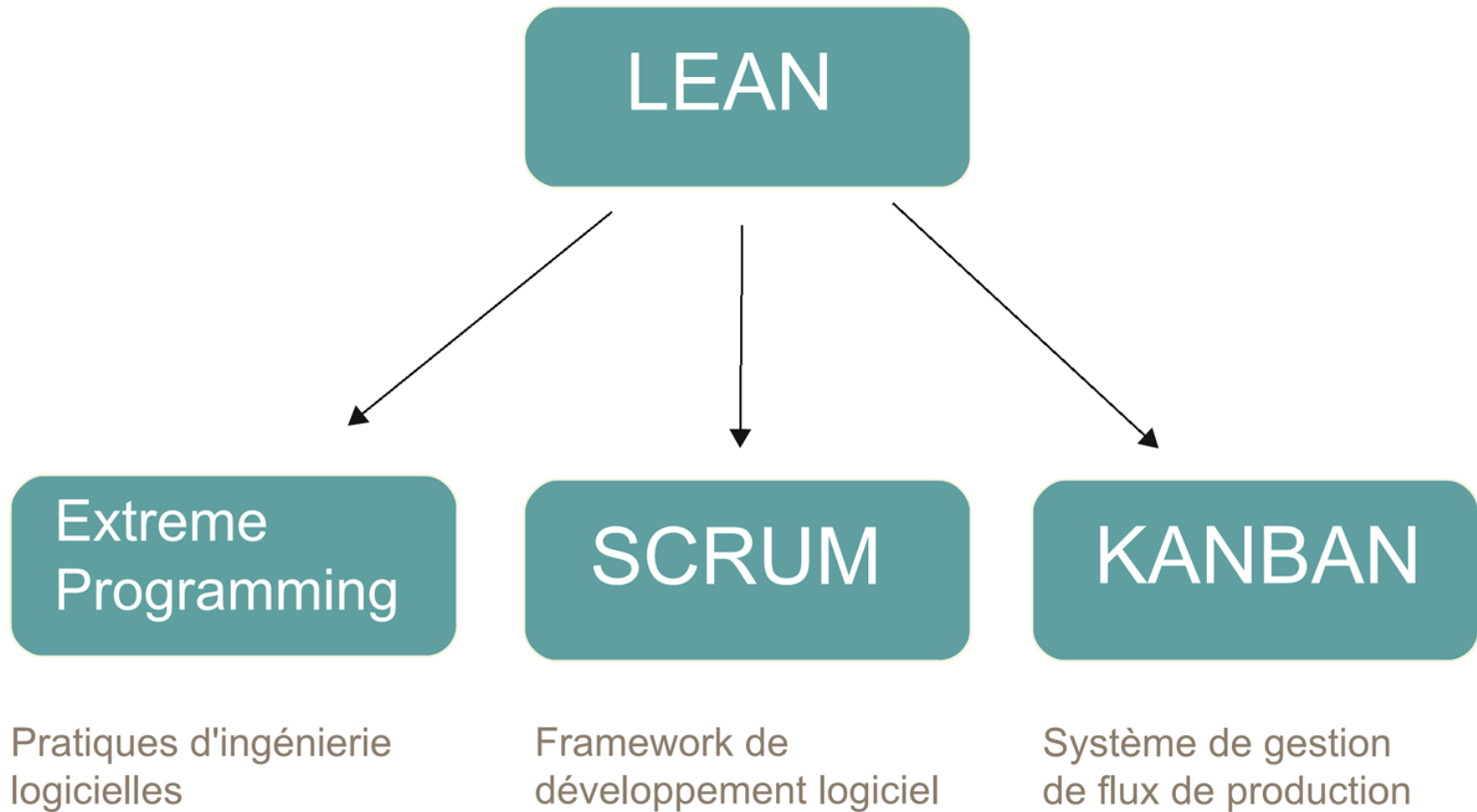
Lean

(Lean software management) - Outils



- Flux tendu
- Élimination de la variation, des gaspillages et de la surcharge.
- Construire de la qualité (intégration dans les processus).
- Management visuel.
- Standardisation créative.
- Formation et mentoring.

Positionnement de Lean



Quel est le cadre Scrum ?



Scrum

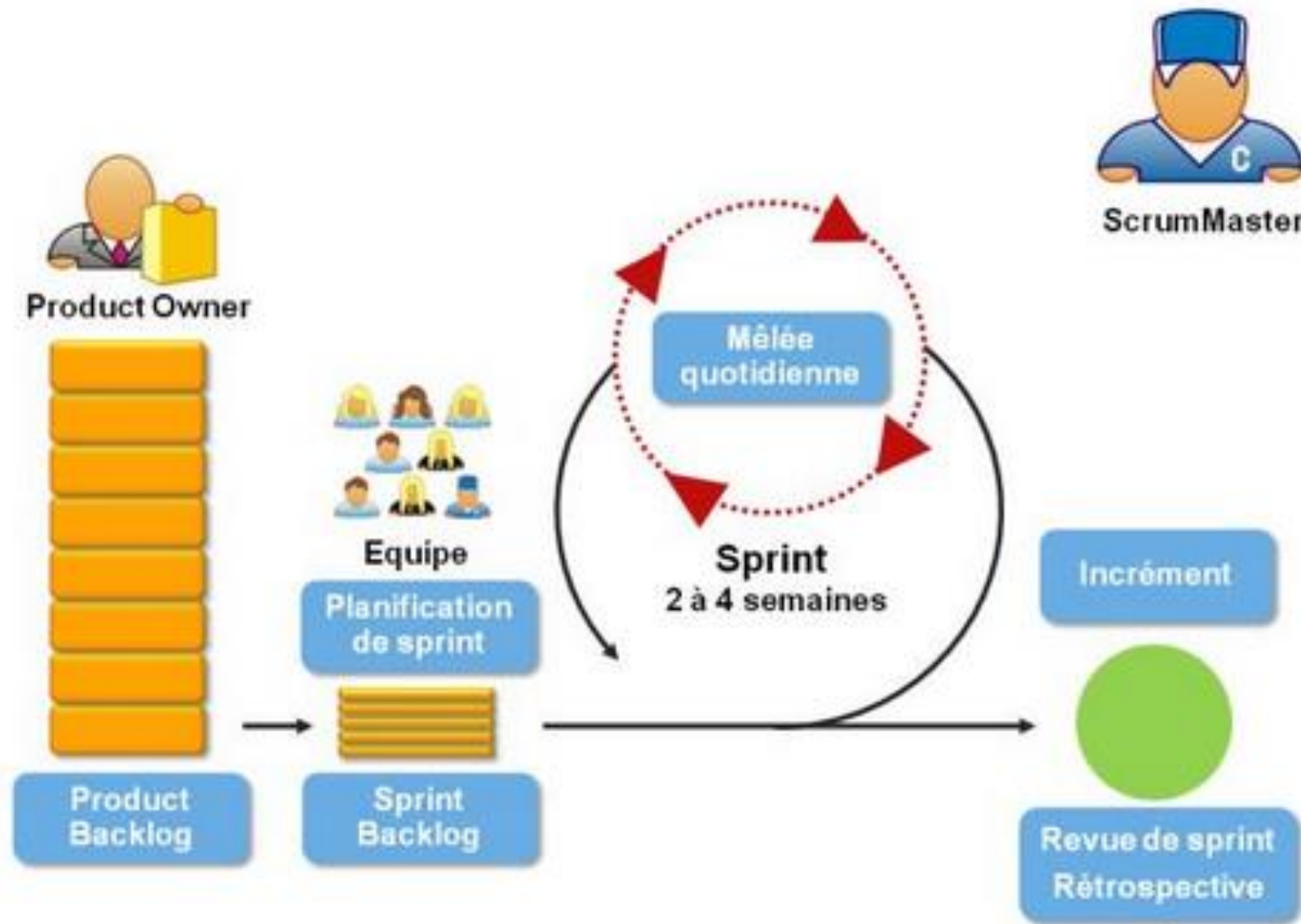


- **Cadre ou « framework » de gestion de projets :**

<https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-French.pdf>

- Développé en 1993 par Ken SCHWABER et par Jeff SUTHERLAND
- Méthode itérative (sprints) et incrémentale
- Équipes de 3-9 personnes
- Mécanismes d'extension possibles (Scrum of Scrums).
- Méthode agile la plus utilisée avec eXtreme Programming

Scrum - Cycle de vie



Scrum - Phases

•Phase Initiale :

- Planning.
- Mise en place d'un backlog (liste des tâches à effectuer).
- Définition de l'équipe.
- Analyse des risques - Budget.

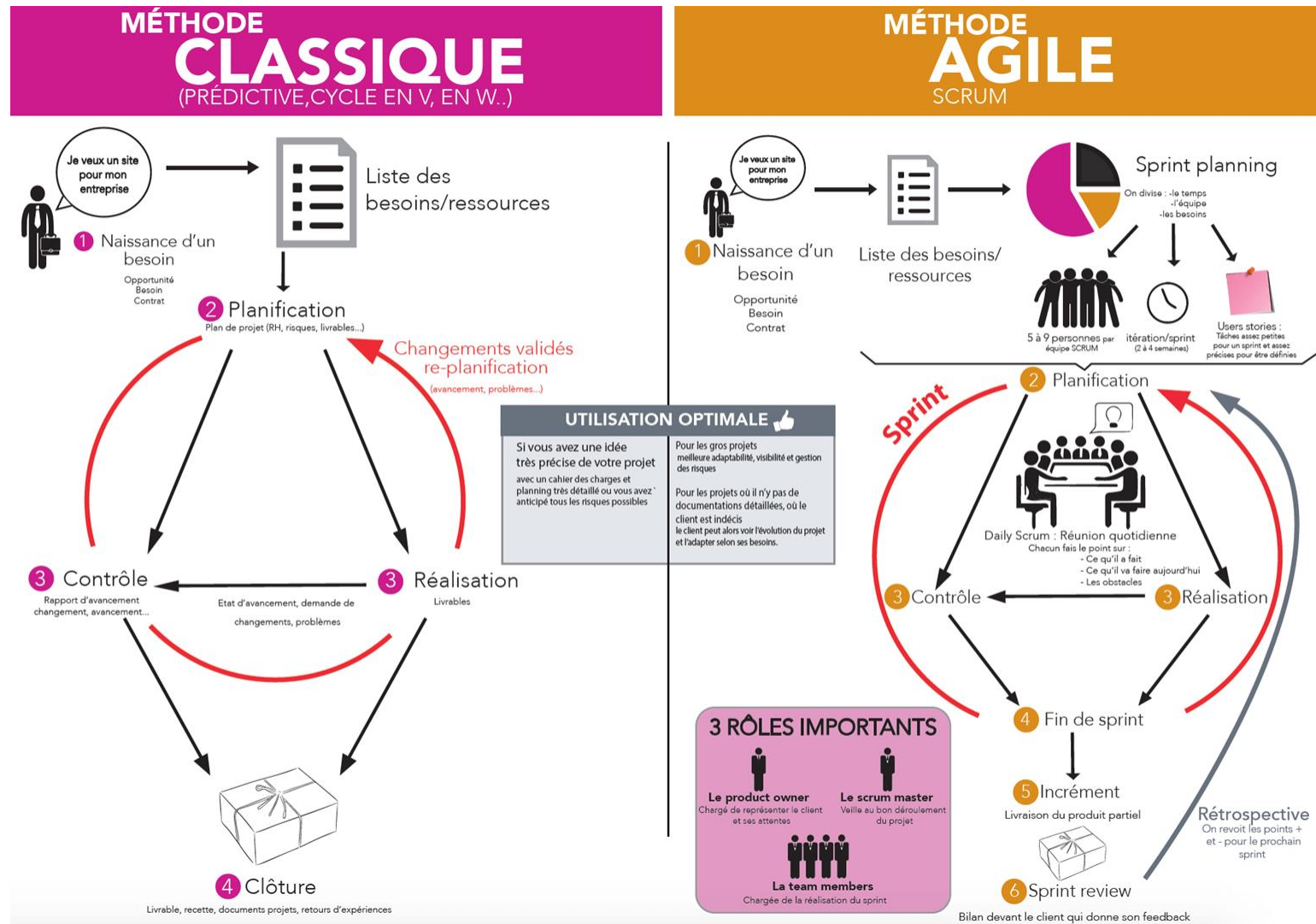
•Sprint :

- 2 à 4 semaines isolées de toute influence extérieure.
- Un sprint backlog est suivi et réalisé.
- Réunion quotidienne Scrum.
- Réunion post-sprint de présentation des résultats.

•Clôture :

- Documentation finale.
- Livraison

Cycle prédictif vs Scrum



Composants

- **3 piliers :**

- - La transparence – L'inspection - L'adaptation

- **Le cadre Scrum :**

- - Des équipes Scrum
- - Des blocs de temps (time-boxes)
- - Des artéfacts
- - Des règles

- **Rôles (équipe) Scrum :**

- - Product Owner
- - Scrum Master
- - Team members

Time boxes

Des blocs de temps :

- Réunion de planification de sprint
(Sprint Planning Meeting)
- Sprint
- Mêlée quotidienne (Daily Scrum Meeting)
- Revue de sprint (Sprint Review/Demonstration Meeting)
- Rétrospective du sprint (Sprint Retrospection Meeting)

Artéfacts

- Carnet de produit (product backlog)
- Carnet de sprint (sprint backlog)
- Graphique de progression (burndown chart)
- Graphique de progression de livraison (release burndown chart)
- Graphique de progression de sprint (sprint burndown chart)

Règles et principes clé



- Des règles lient les blocs de temps, les rôles et les artefacts.
Exemple : seuls les membres de l'équipe peuvent parler pendant la mêlée quotidienne.
- Conforme au manifeste de l'agilité, Scrum met l'accent sur :
 - - Auto-organisation de l'équipe
 - - Pouvoir de décision donné à l'équipe
 - - Délais fixes
 - - Sprint en isolement
 - - Réunions quotidiennes
 - - Livrer un logiciel fonctionnel - démonstration du résultat du sprint
 - - Planning adaptatif

Sprint

Les projets Scrum progressent par une série de sprints

- Équivalents aux itérations XP
- La durée d'un sprint est de 2 à 4 semaines
- Une durée constante apporte un meilleur rythme
- Le produit (partiel) est conçu, codé et testé pendant le sprint
- Planifier la durée pour permettre de différer la prise en compte d'un changement jusqu'au prochain sprint

Planification

- Constitution du backlog produit par le product owner.
- Répartition en sprints et en releases

Responsabilités du PO



- **Expert Métier :**

- Connaît assez le métier pour avoir une vision produit
- - Répond aux questions techniques posées sur le métier par ceux qui créent le produit

- **Avocat de l'utilisateur final :**

- - Décrit le produit en ayant une connaissance des utilisateurs et de son utilisation, afin de servir les deux.

- **Avocat du client :**

- - Connaît les besoins de l'acheteur du produit et sait choisir un ensemble de fonctionnalités présentant une grande valeur ajoutée pour le client.

- **Avocat du métier :**

- - Connaît les besoins de l'organisation qui finance l'élaboration du logiciel et sait choisir un ensemble de fonctionnalités qui servent leurs objectifs.

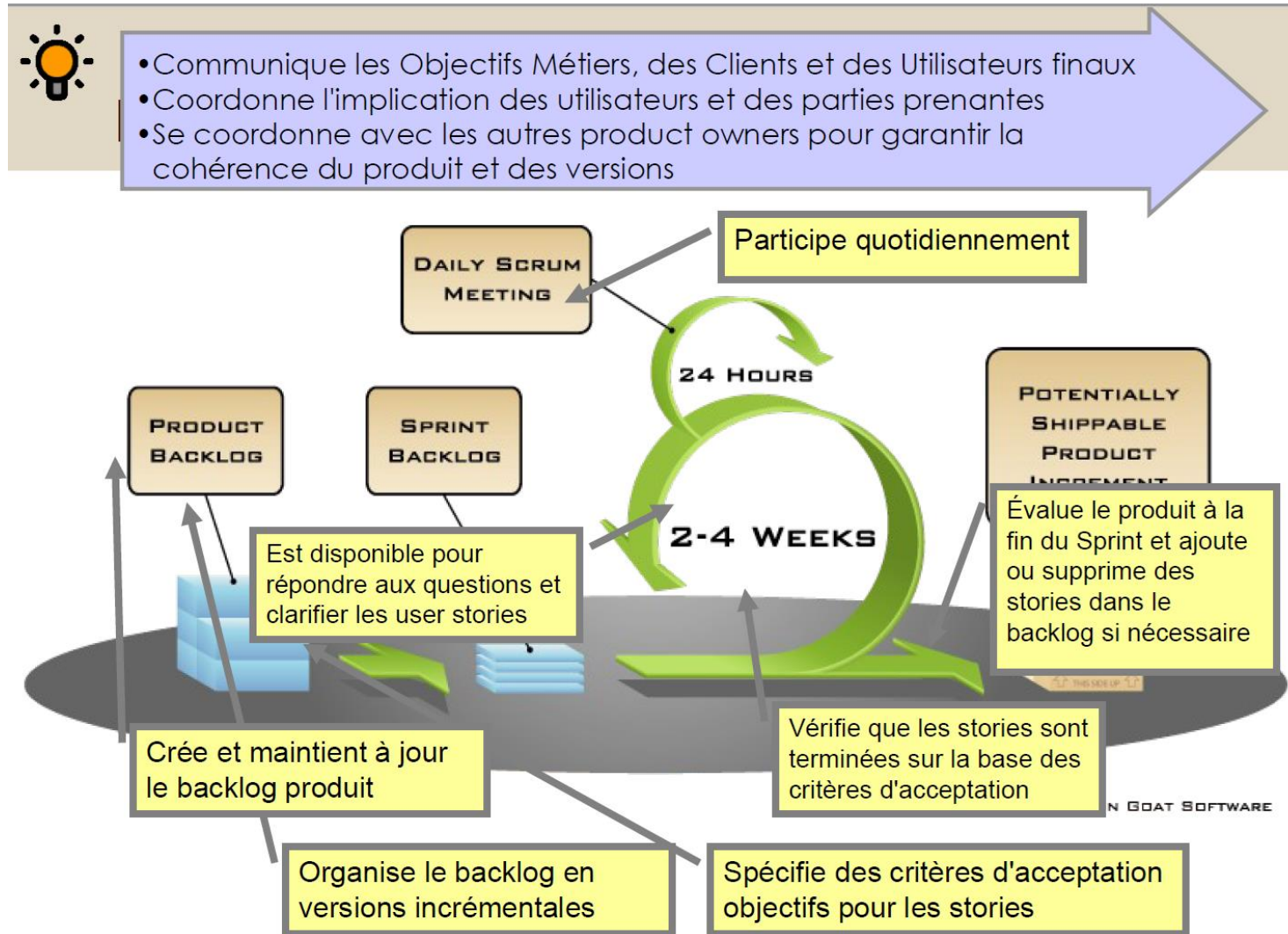
- **Communiquant :**

- - Capable de communiquer sa vision et de différer la spécification d'une fonctionnalité et les choix de conception (Juste à temps).


- **Décideur :**

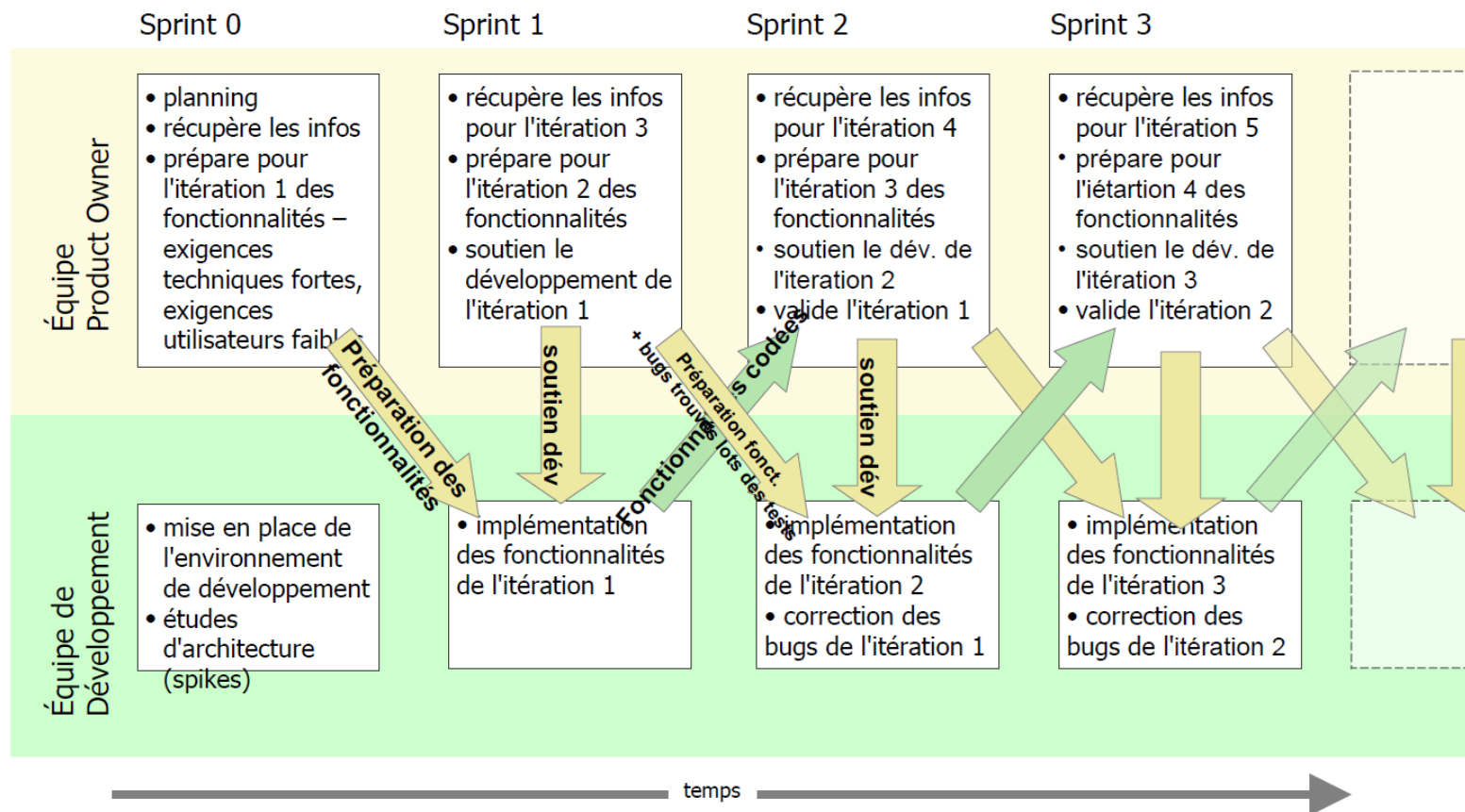
- - Face à une diversité d'objectifs contradictoires et d'opinions, il sait arbitrer et prendre les décisions finales nécessaires

Activités du PO



Activités du PO (2)

 Les fonctionnalités préparées et terminées passent et repassent entre les couloirs



Responsabilités / Rôle

Scrum Master



Représente le management du projet

- Responsable de faire appliquer par l'équipe les valeurs et les pratiques de Scrum
- Résout des problèmes
- S'assure que l'équipe est complètement fonctionnelle et productive
- Facilite une coopération poussée entre tous les rôles et fonctions
- Protège l'équipe des interférences extérieures

Responsabilités / Rôle Development Team

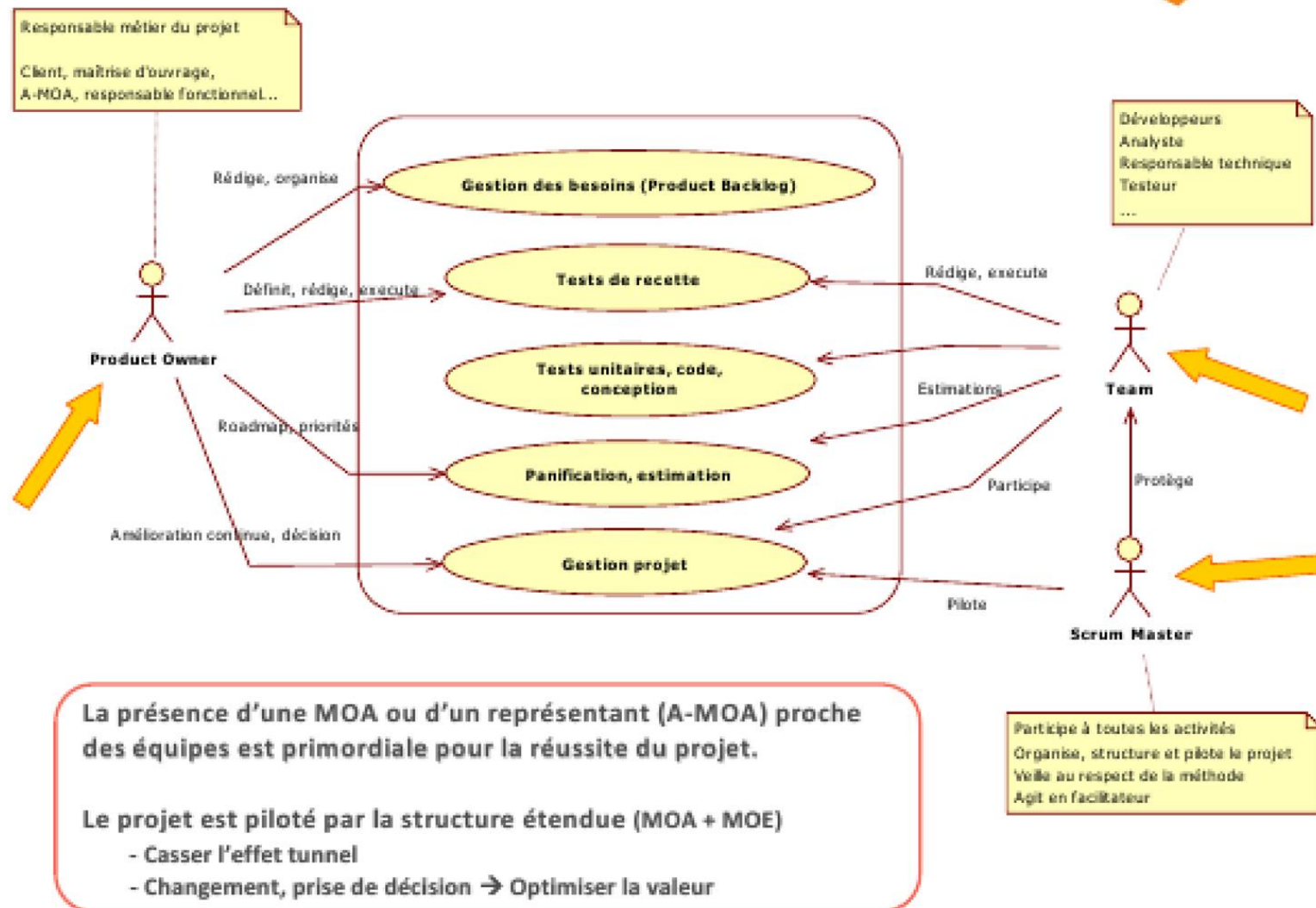


De 3 à 9 personnes

- Regroupant tous les rôles
- Architecte, concepteur, développeur, spécialiste IHM, testeur, etc.
- A plein temps sur le projet, **de préférence**
- Exceptions possibles (administrateur, ...)
- L'équipe s'organise par elle-même
- La composition de l'équipe ne doit pas changer pendant un Sprint

Responsabilités / Rôle

« Récapitulatif »



Les besoins

Avant la 1ère itération

- Le sprint 0
- La formation de l'équipe
- L'environnement de test
- Le product backlog
- La 1ère release planning

Sprint 0

Que devrez vous faire impérativement en Sprint 0 ?

- Partager une Vision claire du projet (ça inclut pas mal de choses)
- Déterminer un plan de release
- Produire un Backlog de produit (« product backlog »), estimé et priorisé (couvrant la release)
- Préparer l'environnement de développement
- Selon les contextes, travailler l'architecture
- Rôder l'équipe ... à tous les niveaux et notamment sur un backlog initial
- Sans oublier vous offrir une belle rétrospective !

Product Backlog

C'est la liste des exigences produit, exprimées si possible sous la forme de User Stories :

« En tant que <rôle>, je veux pouvoir <besoin>, afin de <bénéfice, plus-value>. »

La complexité est estimée par l'équipe SCRUM, par rapport à un item de référence (complexité 1), et sur une échelle issue de la suite de Fibonacci.

La Business Value et la complexité sont utilisés pour calculer la priorité de chaque item du Product Backlog.

Sprint Backlog

Éléments du product backlog sélectionnés pour le sprint.

Chacun s'engage sur du travail qu'il choisit

- Le travail n'est jamais assigné par un autre
- L'estimation du reste à faire est ajustée tous les jours
- N'importe qui peut ajouter, supprimer ou changer le backlog de sprint
- Le travail du sprint émerge progressivement
- Si un travail n'est pas clair, définir une tâche avec plus de temps et la décomposer après.
- MAJ du travail restant quand il devient connu

Sprint planning

L'équipe choisit, à partir du backlog de produit, les éléments qu'elle s'engage à finir.

- Le backlog de sprint est créé
- Durée : 8h (pour un sprint d'un mois)
- Les tâches sont identifiées et estimées (1-16 heures)
- Collectivement, pas seulement par le ScrumMaster
- La conception de haut niveau est abordée

Daily Scrum

Paramètres :

- Tous les jours, 15 minutes, Debout
- Pas fait pour résoudre les problèmes
- Tout le monde est invité
- Seuls les membres de l'équipe de dév. peuvent parler
- Permet d'éviter l'organisation d'autres réunions

Chaque membre répond à 3 questions :

- Qu'as tu fait hier ?
- Que vas-tu faire aujourd'hui ?
- Y a t-il un truc qui cloche ?

Il ne s'agit pas de compte-rendus au Scrum Master

- Ce sont des engagements devant des pairs

Revue de sprint

L'équipe présente ce qu'elle a fait pendant le sprint

- Se fait avec une démo des nouvelles fonctionnalités ou de l'architecture
- Informel
- Préparation < 2h, Durée : 4h/sprint 1mois (~5% de la durée du sprint)
- Pas de slides
- Toute l'équipe participe
- On invite du monde (utilisateurs finaux)

Rétrospective de sprint



Réfléchir régulièrement à ce qui marche et ce qui ne marche pas

- Durée 3h pour un sprint d'un mois
- Fait à la fin de chaque sprint
- Toute l'équipe participe :
 - - Scrum Master
 - - Product Owner
 - - Équipe de développement
- Éventuellement clients et autres intervenants

Scrum of Scrums

- Plusieurs équipes Scrum qui partagent le **même Product Backlog**.
- Facteurs dans la scalabilité :
 - - Type d'application
 - - Taille de l'équipe
 - - Répartition géographique des équipes
 - - Durée du projet
- Scrum a été utilisé pour de nombreux projets de plus de 500 personnes

Faiblesses de la méthode



- Scrum ne donne aucune indication sur les aspects techniques de réalisation.
- Il faut toujours associer à des pratiques de conduite de projet agile, une ingénierie logicielle adaptée. Raison pour laquelle, Scrum peut servir de base dans une organisation agile, mais doit être associée à des méthodes dédiées aux aspects techniques.

Conclusion

- Méthode de gestion de projet – développement logiciel
 - A compléter avec des techniques d'ingénierie logicielle
 - Rien de totalement nouveau
 - Méthode à la mode. Conditions propices nécessaires
 - Expérimentations prometteuses
 - Principal bénéfice : des équipes motivées
-
- TP : Lecture du Scrum Guide et passage du Open Assessment ([scrum.org](https://www.scrum.org))

Plus d'informations sur <http://www.dawan.fr>

**Contactez notre service commercial au
09.72.37.73.73 (prix d'un appel local)**

