Gestion de projet: méthodes et outils
III - méthodes agiles & "scrum"

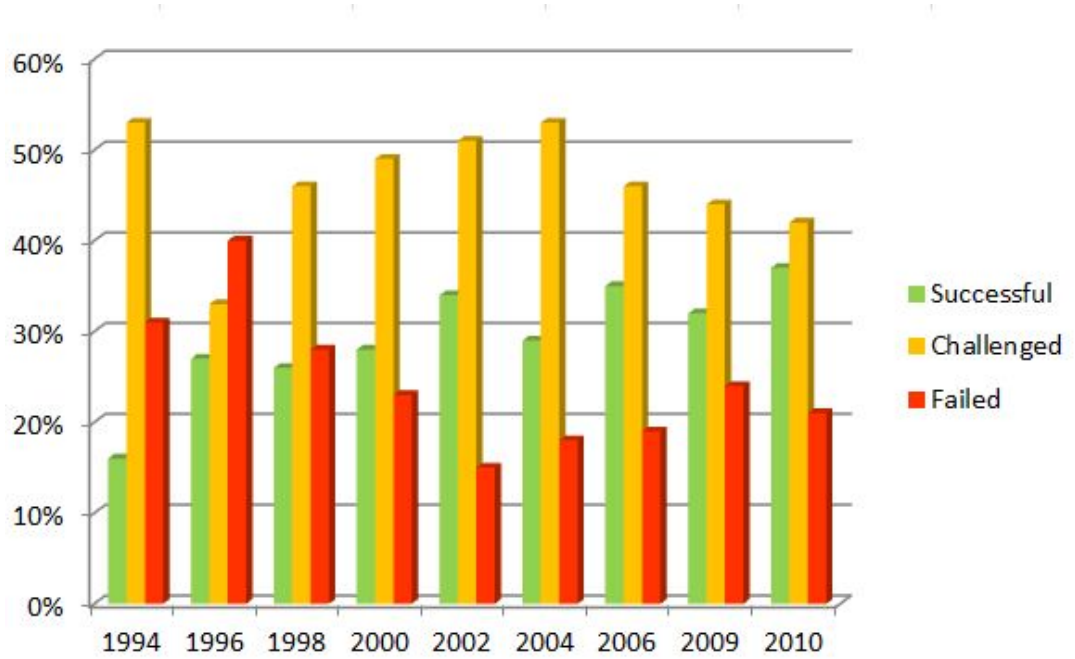
Master-I parcours SSD

Pierre Mahé - bioMérieux & Université de Grenoble-Alpes

Introduction

Motivation : un constat

Etude menée sur les projets logiciel par le Standish Group (Chaos Report) :



- projet "challengé" : terminé et opérationnel, mais plus cher, plus long et moins performant que prévu
- pour 2015 : 29% / 52% / 19%

Motivation : causes admises d'échec

- spécifications inadaptées (incomplètes ou irréaliste)
- manque de ressource
- objectifs mal définis
- planning non réaliste
- manque de communication dans le projet
- non engagement des utilisateurs
- manque de soutien (sponsors)
- manque de compétence
- ...

(chiffres tirés d'une étude menée en Nouvelle-Zélande en 2010 au sein de différents secteurs industriels)

Limites de l'approche "traditionnelle"

Limites fondamentales du découpage définition / réalisation :

Limites de l'approche "traditionnelle"

Limites fondamentales du découpage définition / réalisation :

- difficile de spécifier une bonne fois pour toutes le produit au démarrage du projet
  - spécifications incomplètes et/ou inadaptées
  - peu/pas d'interactions avec le client lors du projet

Limites de l'approche "traditionnelle"

Limites fondamentales du découpage définition / réalisation :

- difficile de spécifier une bonne fois pour toutes le produit au démarrage du projet
  - spécifications incomplètes et/ou inadaptées
  - peu/pas d'interactions avec le client lors du projet
- conduit à de lourds documents de spécifications
  - risque de mé-compréhension de la part de l'équipe projet
  - perte d'efficacité à les rédiger

Limites de l'approche "traditionnelle"

Limites fondamentales du découpage définition / réalisation :

- difficile de spécifier une bonne fois pour toutes le produit au démarrage du projet
  - spécifications incomplètes et/ou inadaptées
  - peu/pas d'interactions avec le client lors du projet
- conduit à de lourds documents de spécifications
  - risque de mé-compréhension de la part de l'équipe projet
  - perte d'efficacité à les rédiger
- difficile de s'adapter au changement
  - e.g., nouvelles fonctionalités

Limites de l'approche "traditionnelle"

Limites fondamentales du découpage définition / réalisation :

- difficile de spécifier une bonne fois pour toutes le produit au démarrage du projet
  - spécifications incomplètes et/ou inadaptées
  - peu/pas d'interactions avec le client lors du projet
- conduit à de lourds documents de spécifications
  - risque de mé-compréhension de la part de l'équipe projet
  - perte d'efficacité à les rédiger
- difficile de s'adapter au changement
  - e.g., nouvelles fonctionalités
- dé-responsabilisation de l'équipe projet

Limites de l'approche "traditionnelle"

Limites fondamentales du découpage définition / réalisation :

- difficile de spécifier une bonne fois pour toutes le produit au démarrage du projet
  - spécifications incomplètes et/ou inadaptées
  - peu/pas d'interactions avec le client lors du projet
- conduit à de lourds documents de spécifications
  - risque de mé-compréhension de la part de l'équipe projet
  - perte d'efficacité à les rédiger
- difficile de s'adapter au changement
  - e.g., nouvelles fonctionalités
- dé-responsabilisation de l'équipe projet
- logiciel : tests d'acception passés trop tardivement

Limites de l'approche "traditionnelle"

Limites fondamentales du découpage définition / réalisation :

- difficile de spécifier une bonne fois pour toutes le produit au démarrage du projet
  - spécifications incomplètes et/ou inadaptées
  - peu/pas d'interactions avec le client lors du projet
- conduit à de lourds documents de spécifications
  - risque de mé-compréhension de la part de l'équipe projet
  - perte d'efficacité à les rédiger
- difficile de s'adapter au changement
  - e.g., nouvelles fonctionalités
- dé-responsabilisation de l'équipe projet
- logiciel : tests d'acception passés trop tardivement

⇒ émergence en 2001 de la notion / du mouvement d'Agilité.

Méthodes agiles : Manifeste de 2001 [1]

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

Ces expériences nous ont amenés à valoriser :

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

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

1. https://agilemanifesto.org/iso/fr/manifesto.html

Méthodes agiles : principes fondateurs [2]

1. Satisfaire le client en livrant tôt et régulièrement des logiciels utiles, qui offrent une véritable valeur ajoutée
2. Accepter les changements, même tard dans le développement
3. Livrer fréquemment une application qui fonctionne
4. Collaborer quotidiennement entre clients et développeurs
5. Bâtir le projet autour de personnes motivées en leur fournissant environnement et support, et en leur faisant confiance
6. Communiquer par des conversations en face à face
7. Mesurer la progression avec le logiciel qui fonctionne
8. Garder un rythme de travail durable
9. Rechercher l'excellence technique et la qualité de la conception
10. Laisser l'équipe s'auto-organiser
11. Rechercher la simplicité
12. A intervalles réguliers, réflechir aux moyens de devenir plus efficace

2. https://agilemanifesto.org/iso/fr/principles.html

Méthode agile : en bref...

Une méthode agile repose sur :

- un dévelopement itératif et incrémental
  - versions intermédiaires avec de + en + de fonctionalités
  - privilégie l'opérationnel à la documentation
- l'acceptation et l'adaptabilité au changement
  - modification des priorités, nouvelles fonctionalités
- une équipe auto-organisée
  - responsabilisation des membres de l'équipe projet

Méthode agile : en bref...

Une méthode agile repose sur :

- un dévelopement itératif et incrémental
  - versions intermédiaires avec de + en + de fonctionalités
  - privilégie l'opérationnel à la documentation
- l'acceptation et l'adaptabilité au changement
  - modification des priorités, nouvelles fonctionalités
- une équipe auto-organisée
  - responsabilisation des membres de l'équipe projet

Pour apporter :

- plus de valeur au clients et aux utilisateurs
- plus de satisfaction dans leur travail à l'équipe projet

Méthode agile : en bref...

Une méthode agile repose sur :

- un dévelopement itératif et incrémental
  - versions intermédiaires avec de + en + de fonctionalités
  - privilégie l'opérationnel à la documentation
- l'acceptation et l'adaptabilité au changement
  - modification des priorités, nouvelles fonctionalités
- une équipe auto-organisée
  - responsabilisation des membres de l'équipe projet

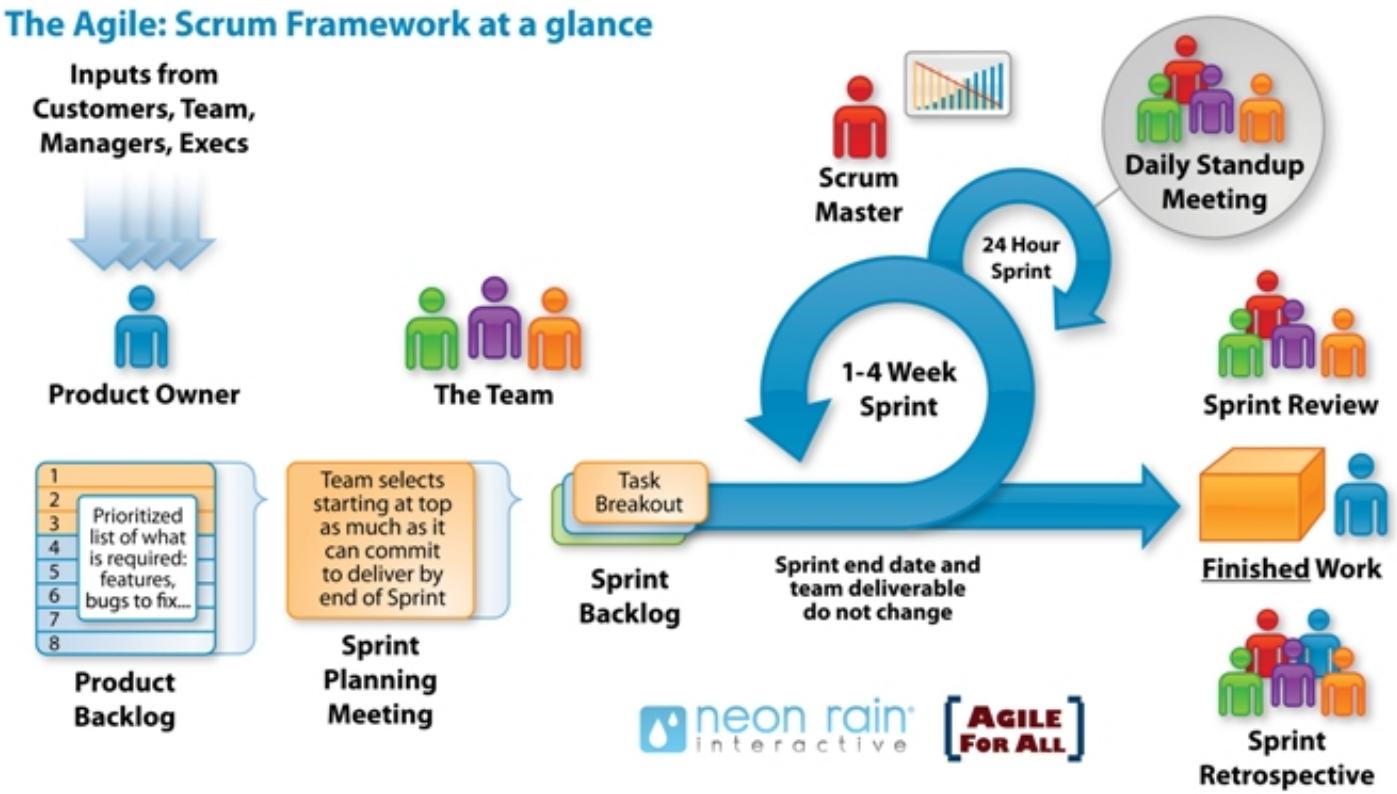
Pour apporter :

- plus de valeur au clients et aux utilisateurs
- plus de satisfaction dans leur travail à l'équipe projet

⇒ `scrum` : une méthode agile incontournable

scrum in a nutshell

scrum : the big picture...

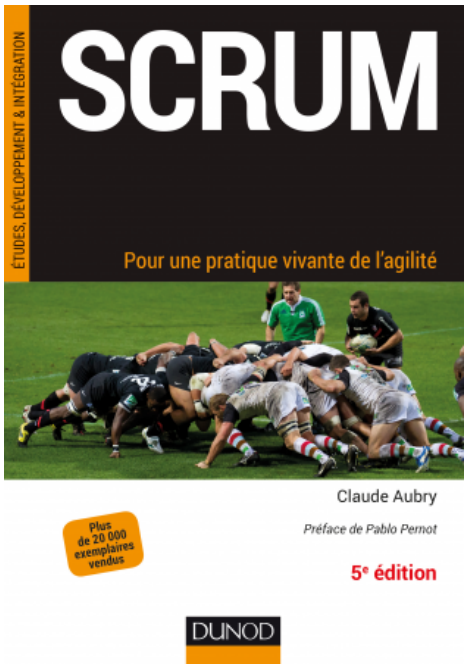


- différents rôles : team, product-owner & scrum-master
- fonctionnalités référencées dans le back-log (les stories)
- fonctionnement en sprints → nouvelle version du logiciel

Ce cours...

Une brève introduction à la méthodologie `scrum`.

Source principale :

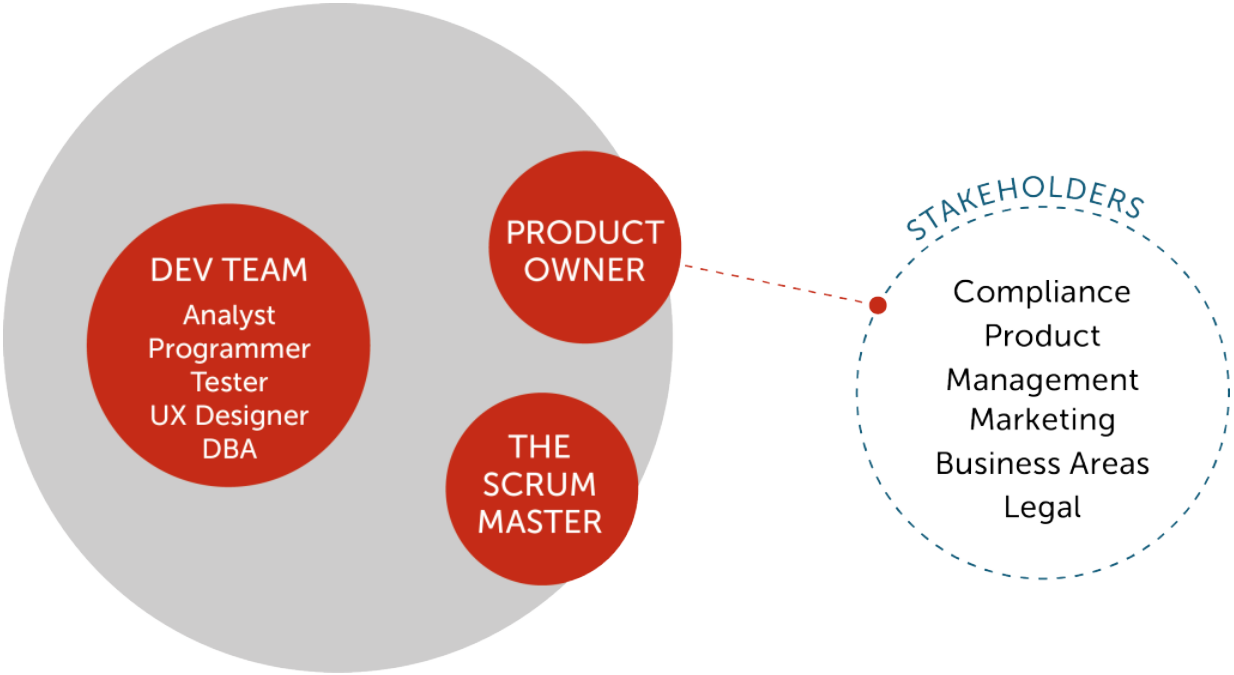


(+ de nombreuses images prises sur internet...)

Product-owner, scrum-master & development team

scrum : différents rôles

Trois rôles principaux [3] :



⇒ principe fondateur : **auto-organisation de l'équipe**

⇒ **pas de chef de projet** : {product-owner + scrum-master}

3. images tirées de https://manifesto.co.uk

Product-Owner

Le Product-Owner :

- est responsable du produit auprès des parties prenantes
- apporte la vision business/client au projet
- définit le contenu du produit, via le backlog
- définit les priorités et l'objectif des différentes releases

Product-Owner

Le Product-Owner :

- est responsable du produit auprès des parties prenantes
- apporte la vision business/client au projet
- définit le contenu du produit, via le backlog
- définit les priorités et l'objectif des différentes releases

Pour cela il doit :

- avoir une bonne connaissance métier
- intéragir avec les parties prenantes pour collecter et formaliser leurs besoins ("user stories" du backlog)
- avoir l'autorité nécessaire pour prendre des décisions
  - i.e., sans se référer à sa hierarchie

Product-Owner

Missons principales :

Product-Owner

Missons principales :

- faire partager la vision
  - pourquoi on fait le produit
  - quelles sont ses fonctionnalités essentielles
  - but d'une release et impacts attendus

Product-Owner

Missons principales :

- faire partager la vision
  - pourquoi on fait le produit
  - quelles sont ses fonctionnalités essentielles
  - but d'une release et impacts attendus
- faire vivre le backlog du projet
  - l'alimenter en fonctionnalités attendues
  - les détailler au bon moment (avant un **sprint**)
  - l'ajuster en fonction du feed-back reçu au long du projet

Product-Owner

Missons principales :

- faire partager la vision
  - pourquoi on fait le produit
  - quelles sont ses fonctionnalités essentielles
  - but d'une release et impacts attendus
- faire vivre le backlog du projet
  - l'alimenter en fonctionnalités attendues
  - les détailler au bon moment (avant un **sprint**)
  - l'ajuster en fonction du feed-back reçu au long du projet
- travailler au quotidien avec l'équipe de développement
  - définir le contenu d'une release et des sprints associés
  - planifier les sprints et évaluer l'atteinte de leurs objectifs
  - répondre à ses questions et lever des points de blocage

Product-Owner

Missons principales :

- faire partager la vision
  - pourquoi on fait le produit
  - quelles sont ses fonctionnalités essentielles
  - but d'une release et impacts attendus
- faire vivre le backlog du projet
  - l'alimenter en fonctionnalités attendues
  - les détailler au bon moment (avant un **sprint**)
  - l'ajuster en fonction du feed-back reçu au long du projet
- travailler au quotidien avec l'équipe de développement
  - définir le contenu d'une release et des sprints associés
  - planifier les sprints et évaluer l'atteinte de leurs objectifs
  - répondre à ses questions et lever des points de blocage
- partager l'avancement avec les parties prenantes
  - animer les réunions de **revues de sprint**

L'équipe de développement

`scrum` repose sur l'auto-organisation de l'équipe :

- elle définit sa manière de travailler et de s'organiser
- elle interagit avec le product-owner pour définir les objectifs d'une release ou d'un sprint (le **quoi**)
- elle définit **(seule)** la manière de les atteindre : tâches et organisation (le **comment**)

⇒ vecteur de responsabilisation et de motivation

L'équipe de développement

`scrum` repose sur l'auto-organisation de l'équipe :

- elle définit sa manière de travailler et de s'organiser
- elle interagit avec le product-owner pour définir les objectifs d'une release ou d'un sprint (le **quoi**)
- elle définit **(seule)** la manière de les atteindre : tâches et organisation (le **comment**)

⇒ vecteur de responsabilisation et de motivation

Pour cela, l'équipe doit :

- être de taille restreinte : 3-10 personnes
- posséder toutes les compétences nécessaires au projet
- être responsable et motivée : clé de l'auto-organisation

Scrum-master

Scrum-master = facilitateur :

- aide l'équipe à s'auto-organiser et à s'améliorer
- protège l'équipe des perturbations extérieures
- "médiateur" entre product-owner et l'équipe
  - e.g., pour definir le contenu des releases et sprints

Scrum-master

Scrum-master = facilitateur :

- aide l'équipe à s'auto-organiser et à s'améliorer
- protège l'équipe des perturbations extérieures
- "médiateur" entre product-owner et l'équipe
  - e.g., pour definir le contenu des releases et sprints

Pour cela il doit :

- maîtriser la méthodologie `scrum`
- veiller à son application
  - e.g., animer la réunion de **scrum quotidien**
- encourager l'équipe à progresser dans son métier
  - e.g., bonnes pratiques d'ingénierie logicielle
  - e.g., animer la réunion de **retrospective de sprint**
- promouvoir collaboration & interactions dans l'équipe

En résumé

- Voice of the customer
- Owns value
- Gathers feedback
- Makes decisions

- Owns the process
- Protects team
- Not the boss
- Facilitator

- Commits to the work
- Swarm on high value tasks
- Has skills to deliver
- Aims to be cross-functional

Stories & backlog

scrum & sprints

Méthodologie agile : développement itératif et incrémental

⇒ développement découpé en releases et sprints

scrum & sprints

Méthodologie agile : développement itératif et incrémental

⇒ développement découpé en releases et sprints

⇒ c'est le product owner qui définit le contenu des releases

- en accord avec l'équipe de développement

scrum & sprints

Méthodologie agile : développement itératif et incrémental

⇒ développement découpé en releases et sprints

⇒ c'est le product owner qui définit le contenu des releases

- en accord avec l'équipe de développement

⇒ Pour cela, il s'appuie sur le backlog et les (user) stories

Backlog

Backlog = outil de spécification du produit

Backlog

Backlog = outil de spécification du produit

Contrairement à un document "classique", le backlog :

Backlog

Backlog = outil de spécification du produit

Contrairement à un document "classique", le backlog :

- se construit / s'affine en continu
  - recueil initial de besoin des utilisateurs
  - feed-back sollicité au fil des différentes versions
  - émergence de nouvelles fonctionalités ou modifications

Backlog

Backlog = outil de spécification du produit

Contrairement à un document "classique", le backlog :

- se construit / s'affine en continu
  - recueil initial de besoin des utilisateurs
  - feed-back sollicité au fil des différentes versions
  - émergence de nouvelles fonctionalités ou modifications
- se traduit en terme de besoin et pas de solution
  - besoin = fonctionnalités
  - l'équipe définira la solution technique

Backlog

Backlog = outil de spécification du produit

Contrairement à un document "classique", le backlog :

- se construit / s'affine en continu
  - recueil initial de besoin des utilisateurs
  - feed-back sollicité au fil des différentes versions
  - émergence de nouvelles fonctionalités ou modifications
- se traduit en terme de besoin et pas de solution
  - besoin = fonctionnalités
  - l'équipe définira la solution technique
- se contruit de manière collaborative
  - entre utilisateurs, product-owner et équipe

Backlog

Backlog = outil de spécification du produit

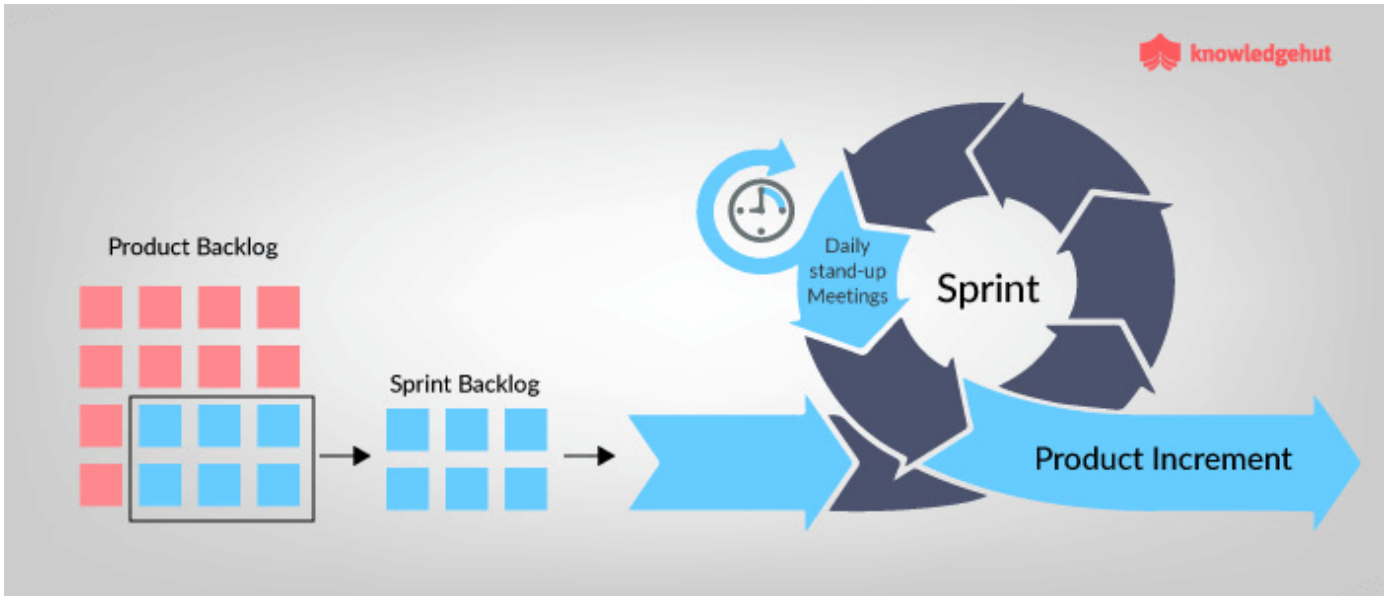
Contrairement à un document "classique", le backlog :

- se construit / s'affine en continu
  - recueil initial de besoin des utilisateurs
  - feed-back sollicité au fil des différentes versions
  - émergence de nouvelles fonctionalités ou modifications
- se traduit en terme de besoin et pas de solution
  - besoin = fonctionnalités
  - l'équipe définira la solution technique
- se contruit de manière collaborative
  - entre utilisateurs, product-owner et équipe

⇒ tient une place centrale dans le projet
⇒ est organisé en "stories"
⇒ est la responsabilité du product-owner

Backlog & sprints

En pratique :



- le backlog contient plusieurs stories
- un sprint va s'atteler à en réaliser un certain nombre
  - liste définie par product-owner + équipe
- les stories du backlog vont être ajustées en continu
  - feedback & ajout, suppression, modification, priorisation

Backlog & stories

Story = élément de base du développement

Backlog & stories

Story = élément de base du développement

Différents types de stories :

- story fonctionnelle = **user-story**
  - développer une nouvelle fonctionalité
- story technique
  - nécessaire au dévelopement (ultérieur) de user-sories
- correction de bug

Backlog & stories

Story = élément de base du développement

Différents types de stories :

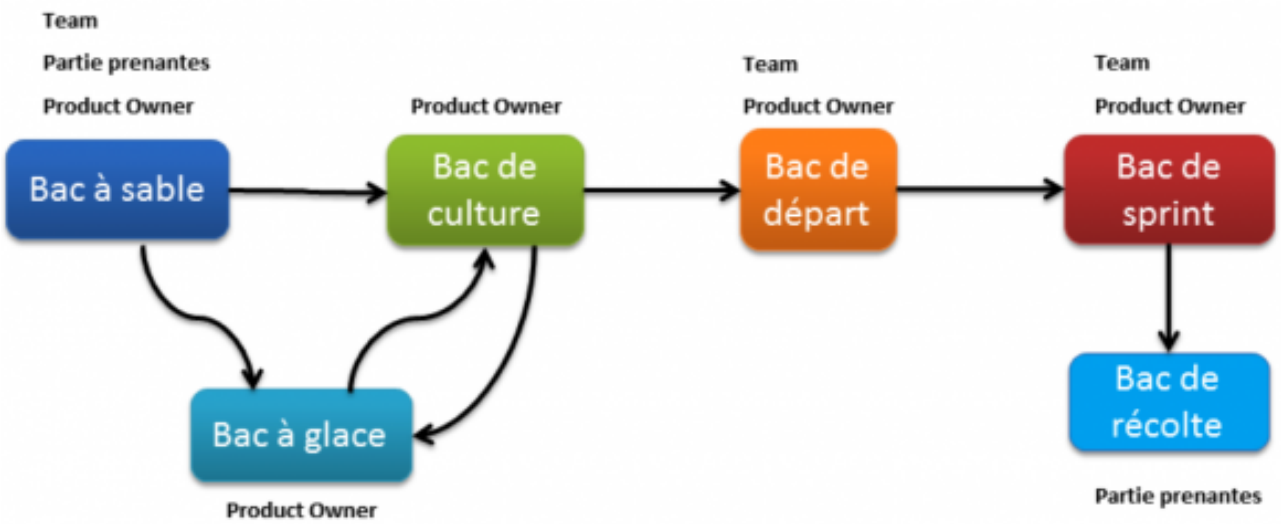
- story fonctionnelle = **user-story**
  - développer une nouvelle fonctionalité
- story technique
  - nécessaire au dévelopement (ultérieur) de user-sories
- correction de bug

Une story est définie par des conditions :

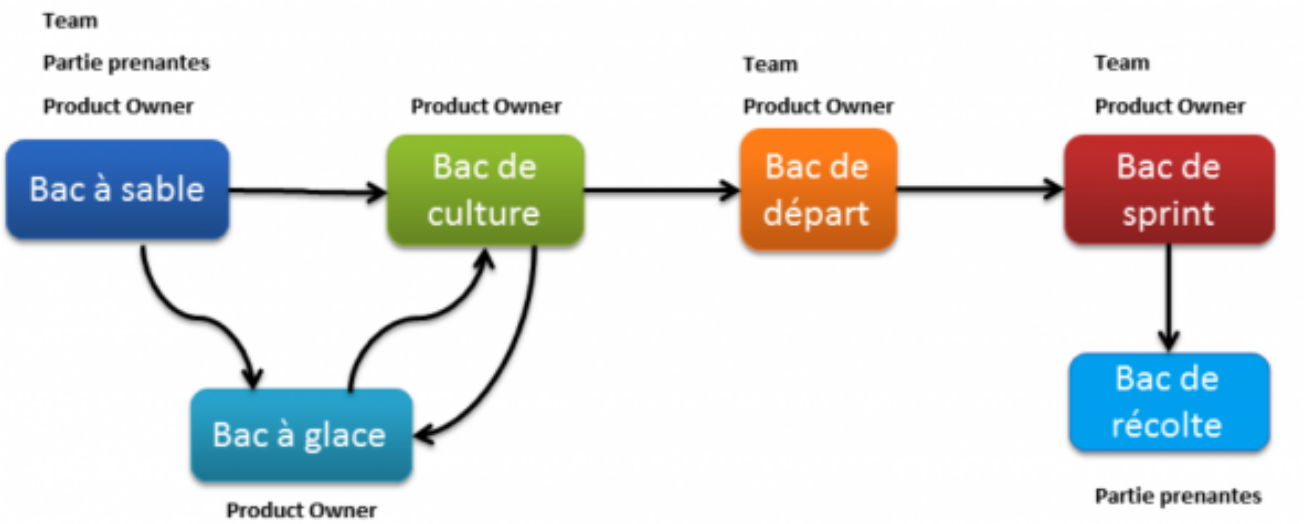
- d'acceptation : comportement attendu
- de finition : niveau de qualité attendu
- de réalisation : compétences et ressources nécessaires

⇒ dialogue PO / équipe pour définir et valider ces conditions

Cycle de vie d'une story

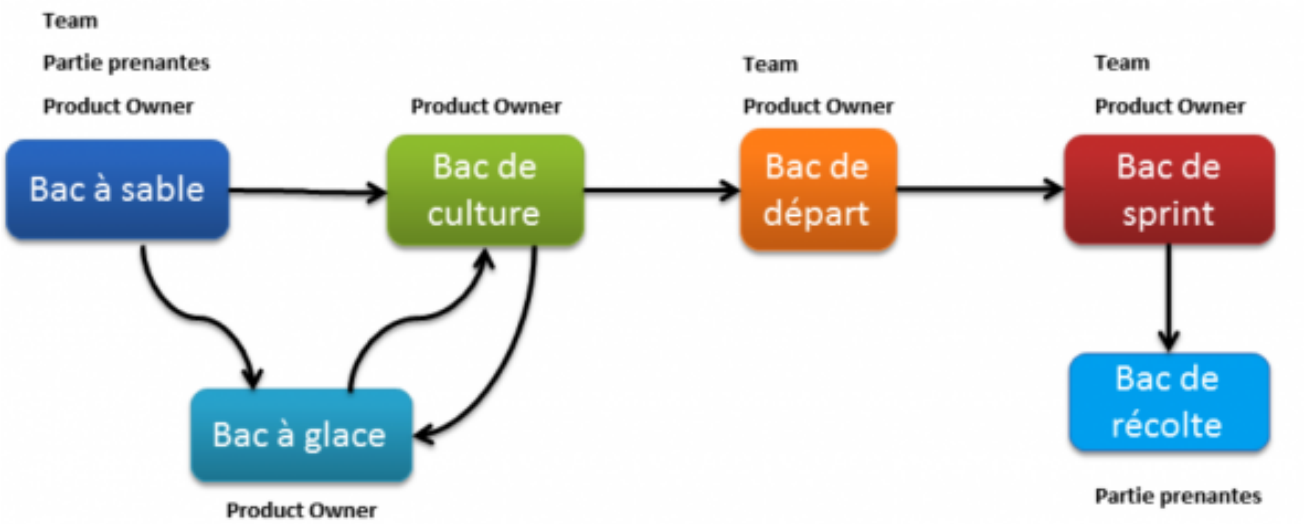


Cycle de vie d'une story



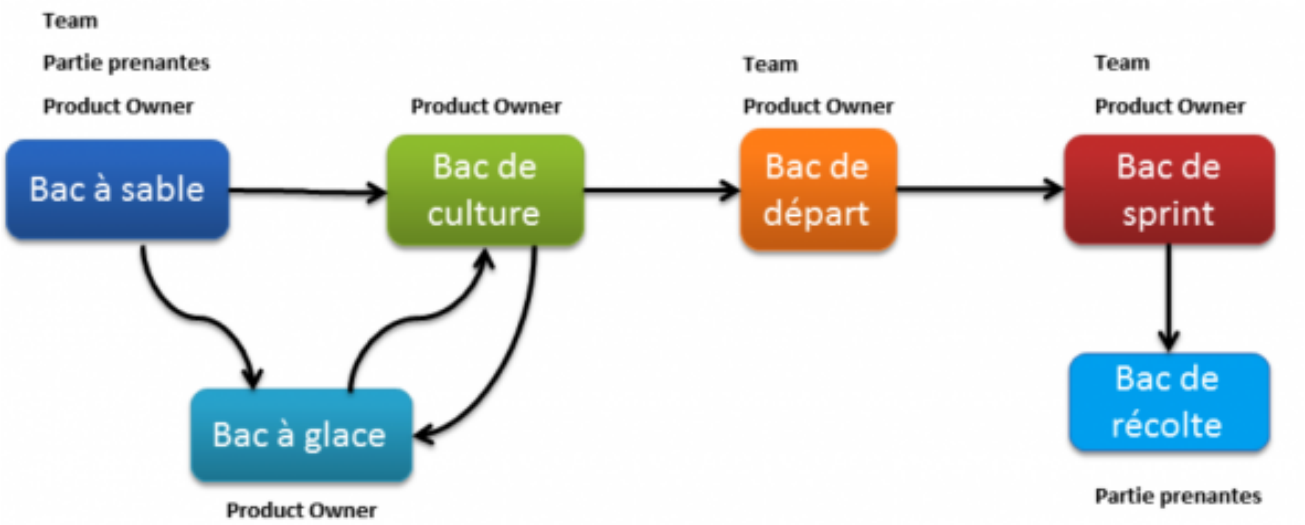
- les stories sont collectées dans le bac à sable
  - contributeurs = team, PO et client / parties prenantes

Cycle de vie d'une story



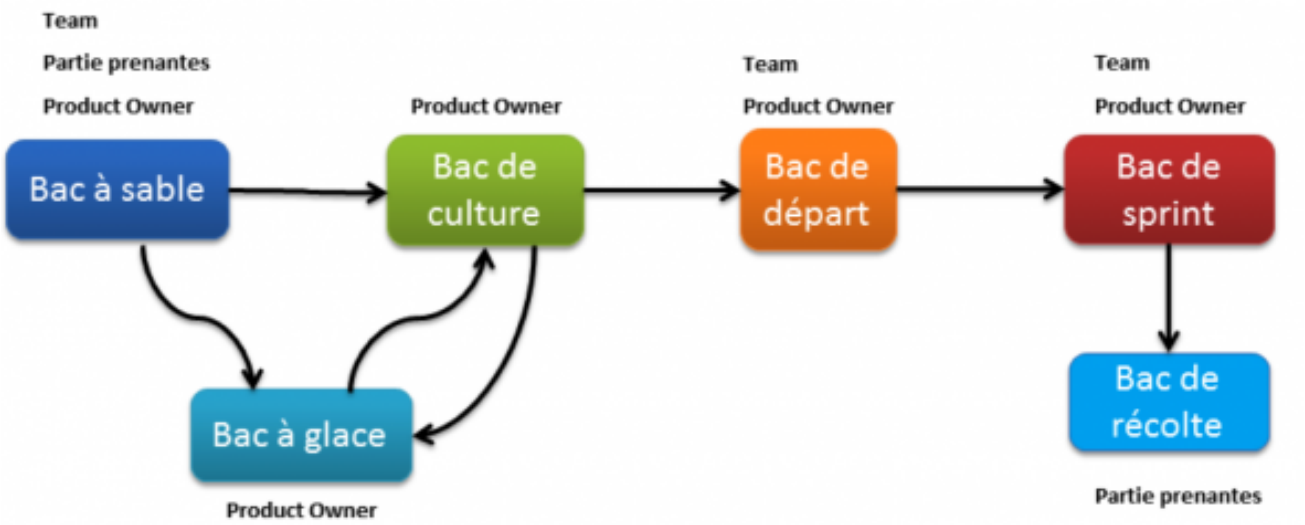
- les stories sont collectées dans le bac à sable
  - contributeurs = team, PO et client / parties prenantes
- le product-owner les "mature" dans le bac de culture
  - ou les place dans le **bac à glace** pour plus tard

Cycle de vie d'une story



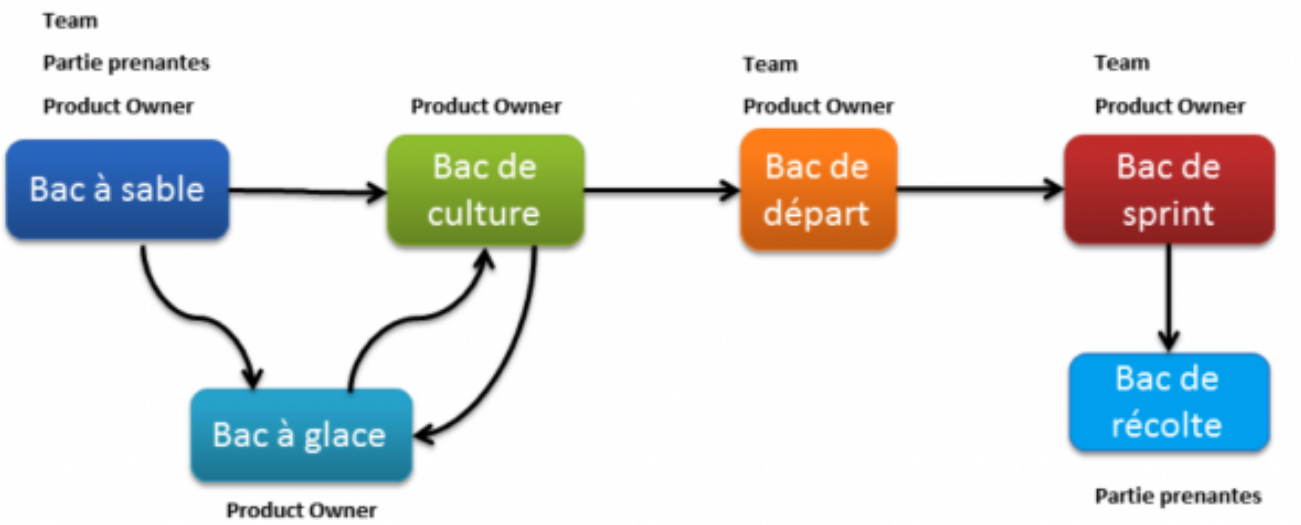
- les stories sont collectées dans le bac à sable
  - contributeurs = team, PO et client / parties prenantes
- le product-owner les "mature" dans le bac de culture
  - ou les place dans le **bac à glace** pour plus tard
- les stories prêtes sont placées dans le bac de départ
  - i.e., si les conditions sont validées par l'équipe

Cycle de vie d'une story



- les stories sont collectées dans le bac à sable
  - contributeurs = team, PO et client / parties prenantes
- le product-owner les "mature" dans le bac de culture
  - ou les place dans le **bac à glace** pour plus tard
- les stories prêtes sont placées dans le bac de départ
  - i.e., si les conditions sont validées par l'équipe
- celles du sprint à venir sont mises dans le bac de sprint

Cycle de vie d'une story



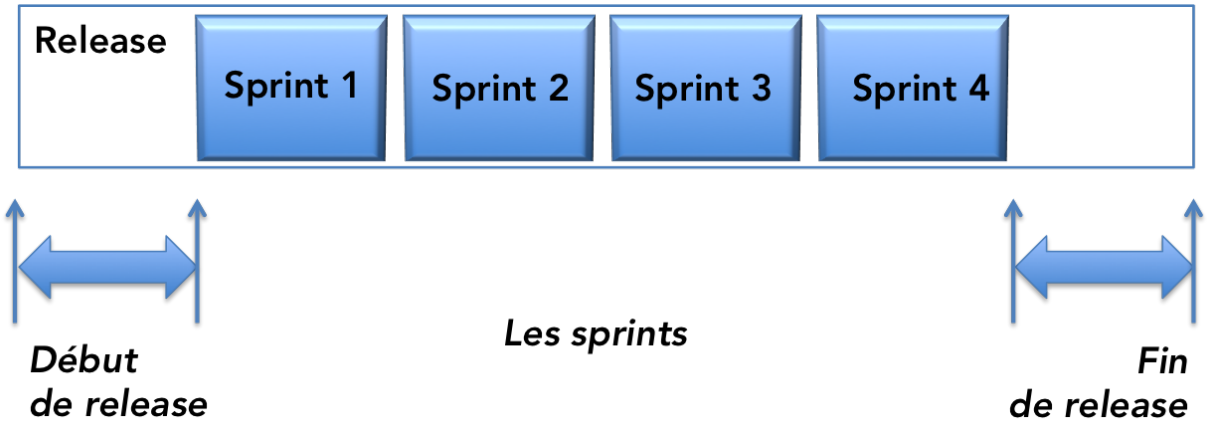
- les stories sont collectées dans le bac à sable
  - contributeurs = team, PO et client / parties prenantes
- le product-owner les "mature" dans le bac de culture
  - ou les place dans le **bac à glace** pour plus tard
- les stories prêtes sont placées dans le bac de départ
  - i.e., si les conditions sont validées par l'équipe
- celles du sprint à venir sont mises dans le bac de sprint
- les stories réalisées sont collectées dans le bac de récolte

Sprints & releases

Releases & sprints

Méthodologie agile : développement itératif et incrémental

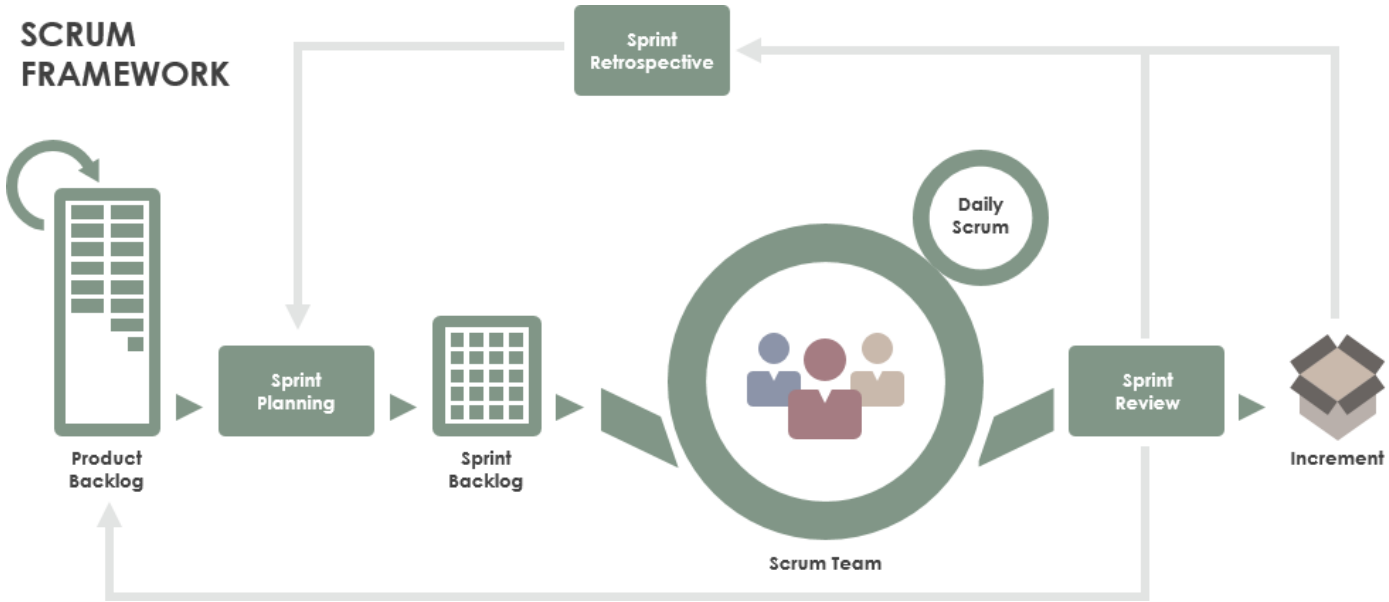
⇒ releases découpées en sprints :



Sprint

Sprint = période de base de développement ~ 2-4 semaines

Différentes étapes :



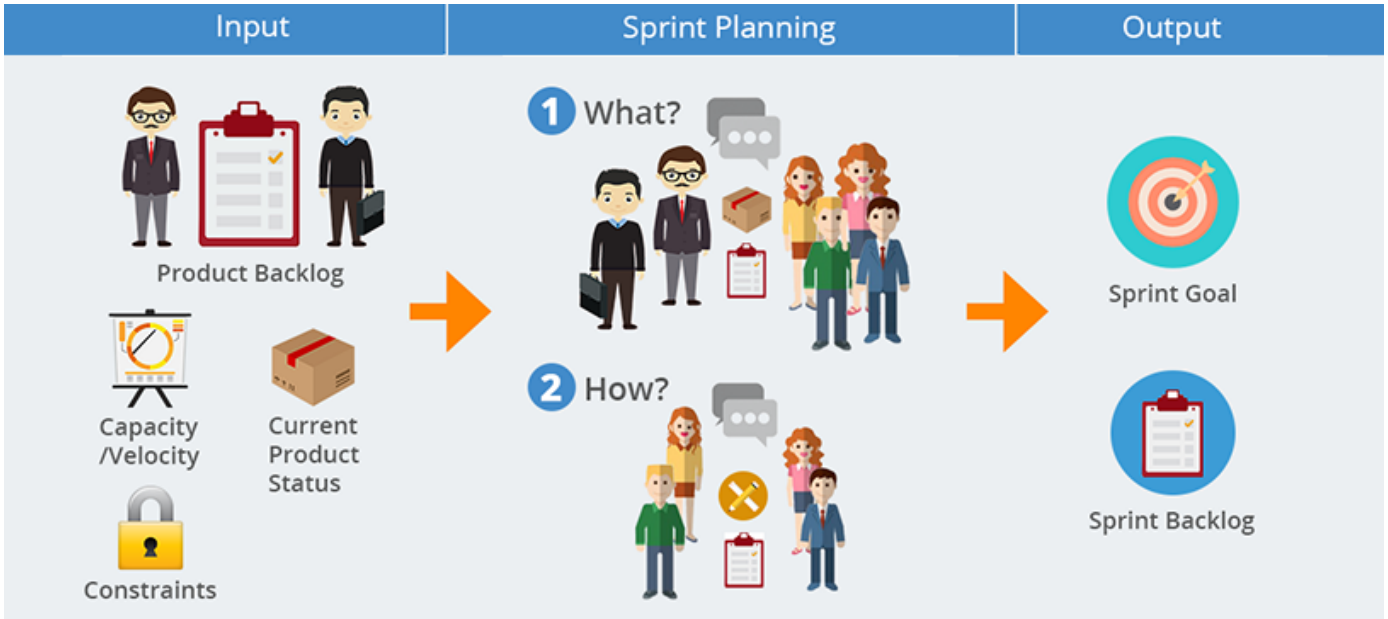
⇒ planning, sprint / développement, review, retrospective

Sprint planning

Objectifs :

1. definir les stories à réaliser dans le sprint (**what ?**)
   - quelles stories passer du bac de départ au bac de sprint
2. découper ces stories en tâches élémentaires (**how ?**)

⇒ action collective mobilisant toute l'équipe (dont le PO)



Sprint planning

En pratique : fait l'objet d'une réunion dédiée

- durée $\sim 2n$ heures pour un sprint de n semaines

Sprint planning

En pratique : fait l'objet d'une réunion dédiée

- durée $\sim 2n$ heures pour un sprint de n semaines

Phase 1 : choix des stories

- le PO les priorise et rappelle leurs conditions
- l'équipe valide leur placement dans le bac de sprint

(nécessite d'"estimer" les stories et la capacité de l'équipe)

Sprint planning

En pratique : fait l'objet d'une réunion dédiée

- durée $\sim 2n$ heures pour un sprint de n semaines

Phase 1 : choix des stories

- le PO les priorise et rappelle leurs conditions
- l'équipe valide leur placement dans le bac de sprint

(nécessite d'"estimer" les stories et la capacité de l'équipe)

Phase 2 : définition des tâches

- l'équipe de développement découpe la story en tâches
- le PO aide (si besoin) à spécifier / valider leurs objectifs

Sprint planning

En pratique : fait l'objet d'une réunion dédiée

- durée $\sim 2n$ heures pour un sprint de n semaines

Phase 1 : choix des stories

- le PO les priorise et rappelle leurs conditions
- l'équipe valide leur placement dans le bac de sprint

(nécessite d'"estimer" les stories et la capacité de l'équipe)

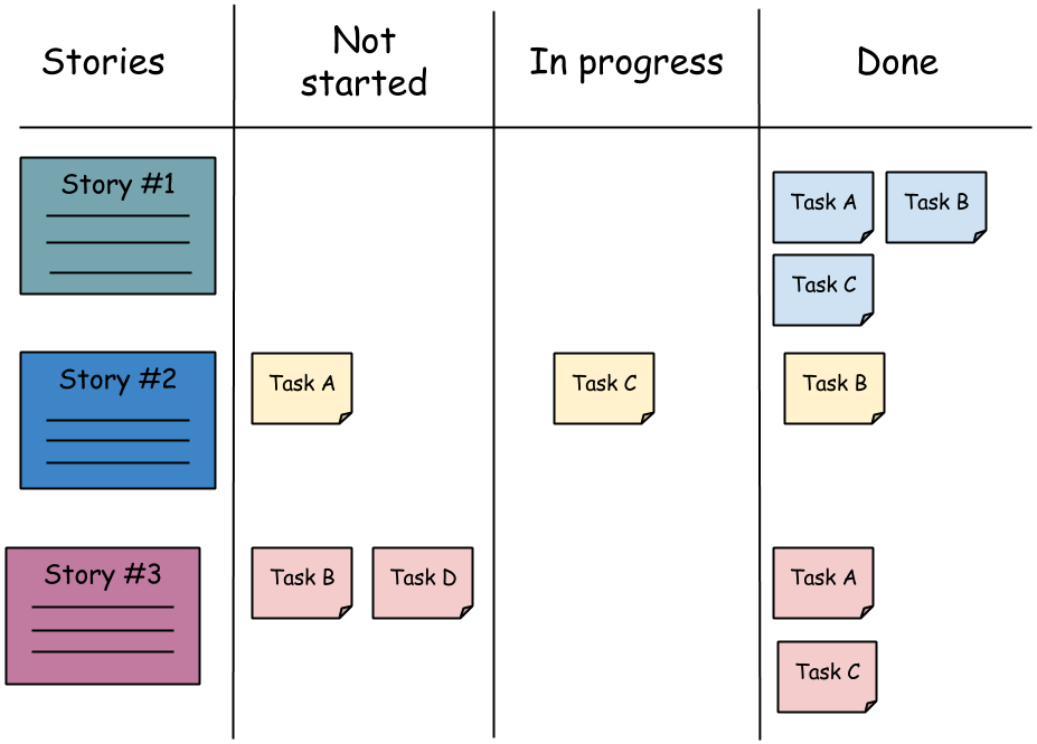
Phase 2 : définition des tâches

- l'équipe de développement découpe la story en tâches
- le PO aide (si besoin) à spécifier / valider leurs objectifs

⇒ à l'issue du planning : on dispose du backlog du sprint

Sprint planning

Backlog du sprint (sprint backlog ou scrum task-board) :



- initialement, toutes les tâches sont "not started"

⇒ outil principal du suivi et de l'organisation du sprint

Sprint & "daily scrum"

A l'issue du planning...le développement débute !

Sprint & "daily scrum"

A l'issue du planning...le développement débute !

Pour cela, les membres de l'équipe

- s'attribuent les tâches les unes après les autres
- renseignent leur avancement dans le sprint backlog

⇒ tenue d'une réunion quotidienne, le **daily scrum meeting**

Sprint & "daily scrum"

A l'issue du planning...le développement débute !

Pour cela, les membres de l'équipe

- s'attribuent les tâches les unes après les autres
- renseignent leur avancement dans le sprint backlog

⇒ tenue d'une réunion quotidienne, le **daily scrum meeting**

Objectif :

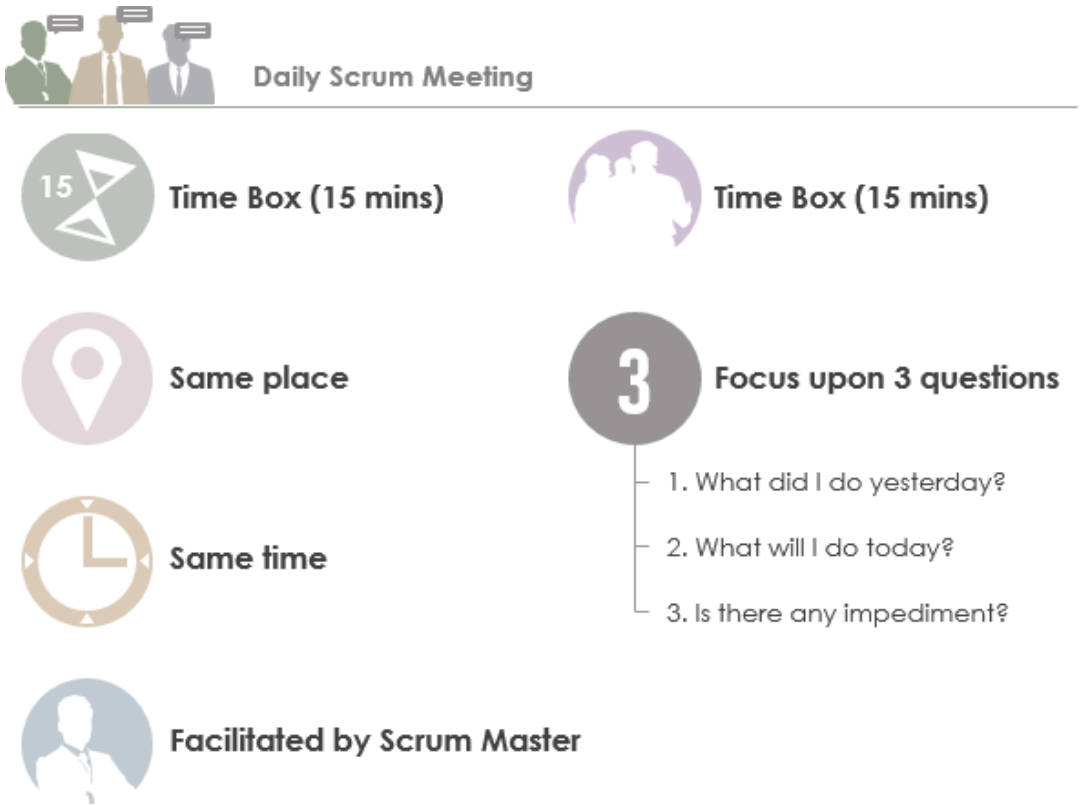
1. partager l'avancement au quotidien
2. identifier (et résoudre) les points de blocage

⇒ cérémonial important de la méthode scrum

Sprint & "daily scrum"

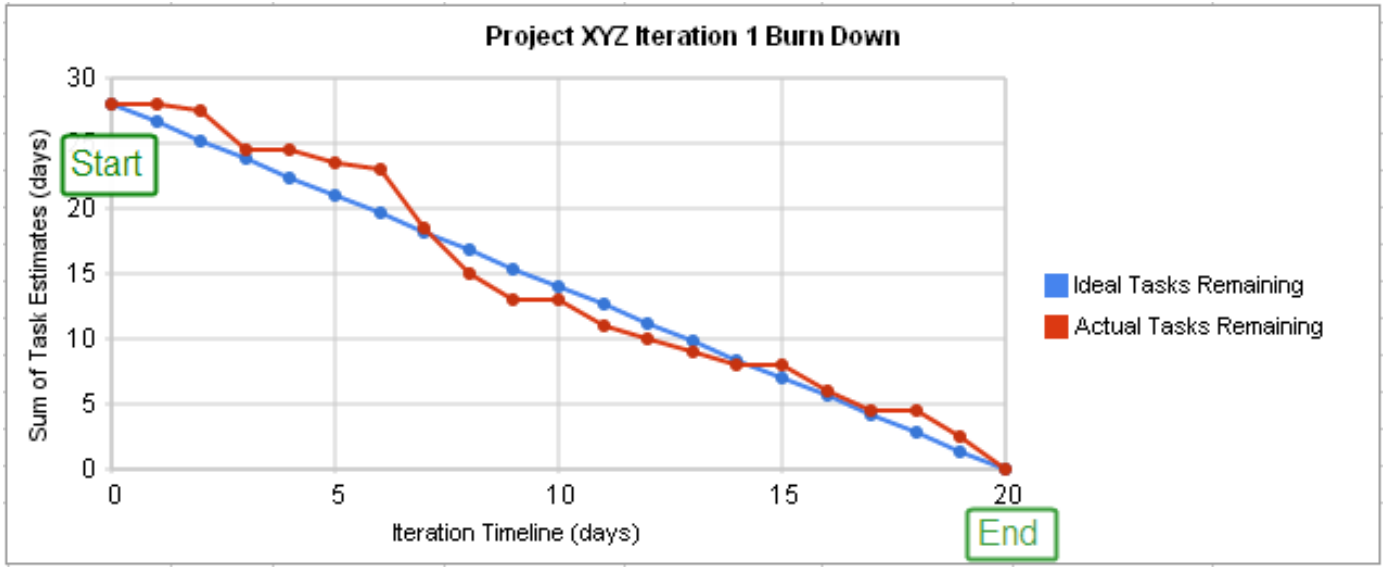
En pratique :

- organisée / animée par le scrum-master
- format court, informel et récurrent (lieu, horaire, durée)
- chacun répond à 3 questions



Sprint & "daily scrum"

Deuxième outil de suivi classique : le burndown-chart



- complété au quotidien
- vue en temps réel du respect du planning
- peut s'exprimer en nombre de tâches ou de "points"
  - 1 tâche = +/- de points en fonction de sa charge

Après le sprint...

A l'issue du sprint, deux réunions :

- sprint review : inspection des résultats
- sprint retrospective : inspection du "process"

En pratique :

- se tiennent à la suite l'une de l'autre
- durent entre 2-4 heures chacune

Sprint review

Objectif : inspecter le produit

- vérifier le bon fonctionnement des "stories" réalisées
- faire une démo aux clients & parties prenantes
- solliciter le feed-back et ajuster le backlog
  - nouvelles stories, nouvelles priorités, ...

Sprint review

Objectif : inspecter le produit

- vérifier le bon fonctionnement des "stories" réalisées
- faire une démo aux clients & parties prenantes
- solliciter le feed-back et ajuster le backlog
  - nouvelles stories, nouvelles priorités, ...

En pratique :

- organisée / animée par le product-owner
- audience plus large que l'équipe projet
  - interactions avec les **utilisateurs**
- dure 2-4 heures (selon durée du sprint)

Sprint review

Déroulement typique :

1. rappel des objectifs du sprint
2. réalisation d'une démonstration
3. product-owner & utilisateurs valident les stories
   - résultats concrets vs conditions d'acceptation
4. collecter le feed-back des utilisateurs
5. actualiser le backlog
   - nouvelles stories, nouvelles priorités, meilleure estimation de leur charge, ...

⇒ permet de définir / ajuster les objectifs du sprint suivant

Sprint retrospective

Objectif : inspecter le "process"

- identifier ce qui a bien / mal marché
- améliorer en continu les performances de l'équipe
  - en termes de **métier** et d'**efficacité**

Sprint retrospective

Objectif : inspecter le "process"

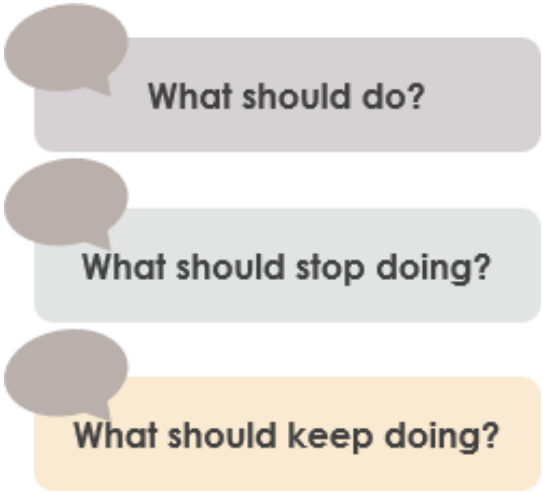
- identifier ce qui a bien / mal marché
- améliorer en continu les performances de l'équipe
  - en termes de **métier** et d'**efficacité**

En pratique :

- organisée / animée par le scrum-master
- audience = équipe projet
  - dev-team + scrum-master + **product-owner**
- dure 1-2 heures (selon durée du sprint)

Sprint retrospective

Chaque membre de l'équipe se pose trois questions :



- $\neq$ axes : pratiques scrum, environnement, outils / métier
- le scrum-master aide à formaliser les retours en actions
- contribue à avoir une équipe plus soudée et motivée
  - remontée points positifs + discussion ouverte sur ce qui peut être amélioré

Conclusion

Méthodologie agile

Principes clés :

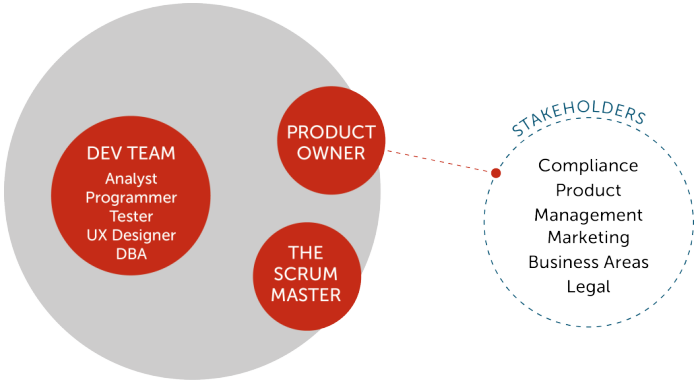
- Développement itératif et incrémental
- Livrer fréquemment un code / produit fonctionnel
- Collaboration régulière équipe projet & client / utilisateur
- Acceptation du changement
- Equipe auto-organisée
- Recherche de l'excellence technique

⇒ rôle important dans l'industrie du développement logiciel

Méthodologie scrum

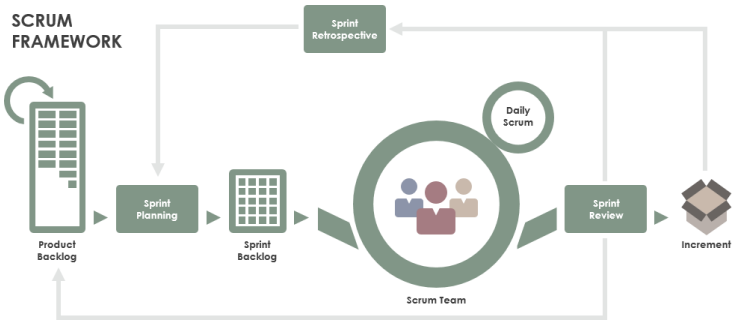
Rôles :

- product-owner
- scrum-master
- équipe de développement



Cérémonial :

- sprint planning
- daily-scrum
- sprint review
- retrospective



Artefacts (principaux) :

- product backlog
- sprint backlog
- burndown chart

Méthodologie scrum

En résumé :