Initiation aux méthodes agiles

01/2020

Objectifs du cours

- Connaître des méthodes agiles et leurs fonctionnements
- Connaîtres des pratiques “agiles” et leurs apports

https://quizizz.com/join?gc=236700

Les principales méthodes agiles

- eXtreme programming - 1990
- Kanban - 1960
- Scrum - 2009
- Rational Unified Process - RUP - 1999

Les principales méthodes agiles

- Rapid Application Development - RAD - 1984
- Adaptive Software Development - ASD - 1995
- Crystal - 1990
- Dynamic Systems Development Method - DSDM - 1994
- Feature Driven Development - FDD - 2002

Kanban

Origine

Méthode Kanban

Les objectifs de Kanban est de minimiser:

- les WIP ("Work-In-Progress", travail en cours),
- le stock ("inventory"),

Méthode Kanban

4 principes :

1. Commencez par ce que vous faites actuellement
2. Acceptez d’appliquer des changements progressifs
3. Respectez le processus, les rôles, les responsabilités et les titres
4. Leadership à tous niveaux

Méthode Kanban

5 bonnes pratiques :

1. Visualisation
2. Mettre des limites
3. Gestion des flux
4. Avoir des règles explicites dans l'équipe
5. Amélioration continue

Méthode Kanban

Les apports

- Produire juste à temps
- Par la visualisation mettre en évidence tout point de blocage
- Favorise la collaboration dans l’équipe
- Responsabilise les membres de l’équipe

Méthode Kanban

TODO	DEV	TEST	DEPLOY
1			
2			
3			
4			

Méthode Kanban

H

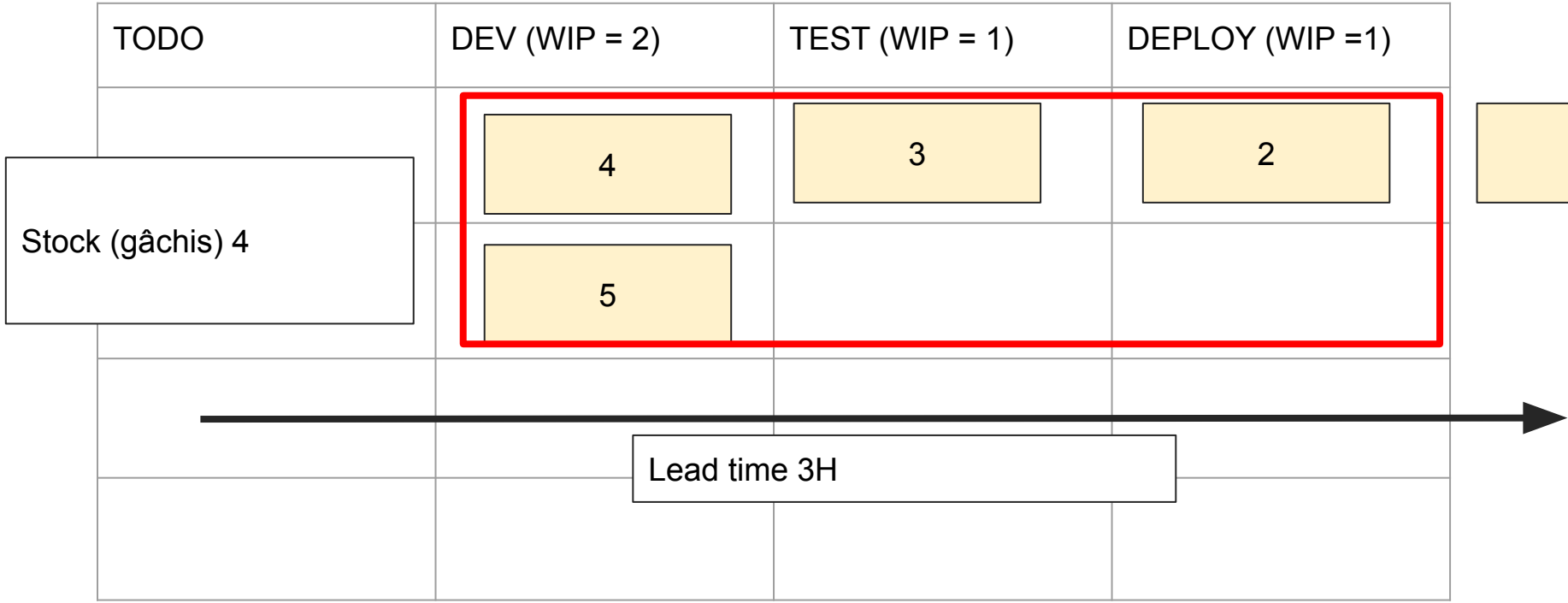
TODO (WIP¨= 3)	DEV (WIP = 2)	TEST (WIP = 1)	DEPLOY (WIP =1)
3	1		
4	2		
5			

Méthode Kanban

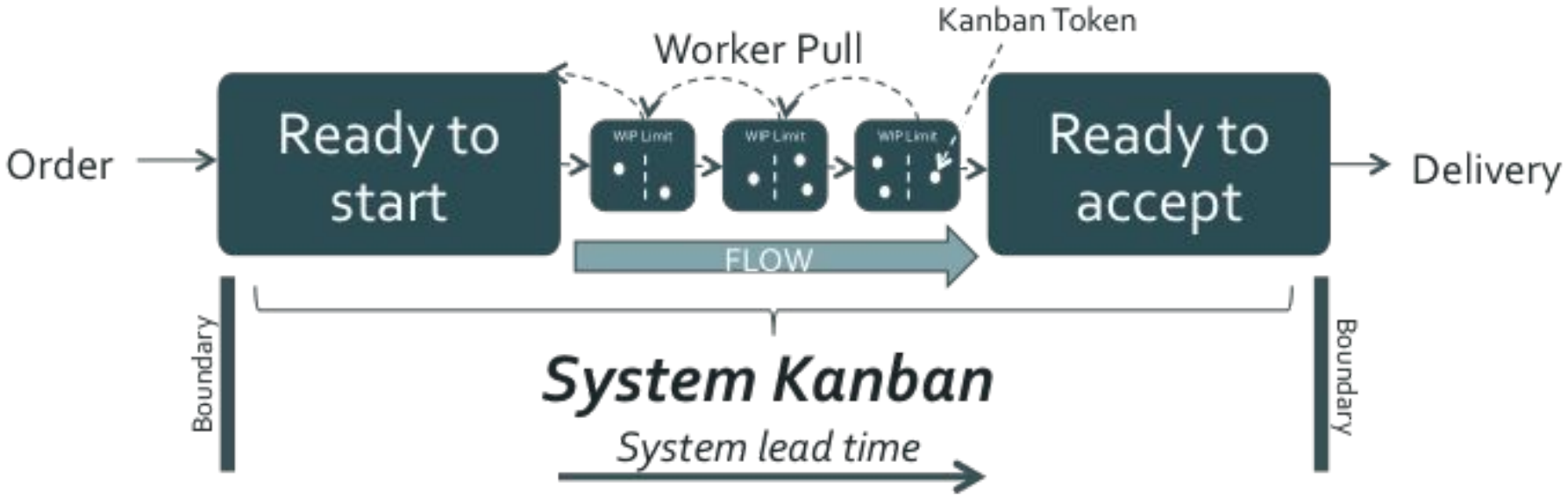
H

TODO (WIP¨= 3)	DEV (WIP = 2)	TEST (WIP = 1)	DEPLOY (WIP =1)
5	3	2	1
	4		

Méthode Kanban



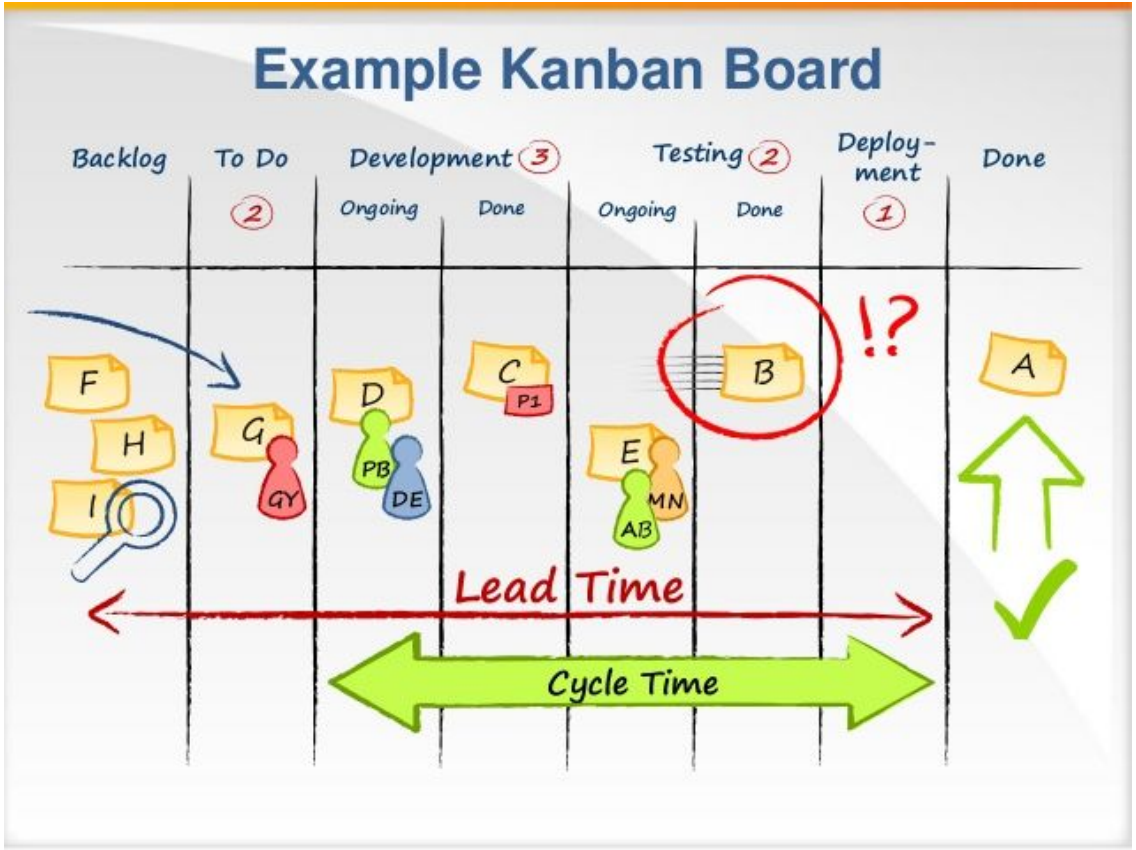
Méthode Kanban



Quelques exemples

La méthode Kanban apporte au monde de l'agilité un focus particulier sur l'amélioration des processus et la fluidité des flux de production. Ses artefacts les plus connus sont le tableau kanban et les limites du travail en cours.

Kanban


Example Kanban Board
Backlog
To Do
2
Development
3
Ongoing
Done
Testing
2
Ongoing
Done
Deploy-ment
1
Done
F
H
I
G
GY
D
PB
DE
C
P1
E
AB
MN
B
!?
A
Lead Time
Cycle Time

Lean

Lean

- Lean = “sans gras”, au plus juste.
- Lean est une approche et non une méthode
- Approche de gestion de production chez Toyota
- S’applique aux développements informatiques agiles, mais aussi aux administrations, aux organisations
- Se concentre sur l’élimination du gaspillage

Les principes du Lean

1. Éliminer le gaspillage
2. Améliorer l’apprentissage.
3. Décider aussi tard que possible (mais pas trop tard ....)
4. Livrer aussi vite que possible
5. Donner le pouvoir à l’équipe
6. Intégrer la qualité dès la conception
7. Considérer le produit dans sa globalité

lean startup

Le ***lean start up*** est une approche spécifique du démarrage d'une activité économique et du lancement d'un produit.

« Validated learning `(en)` » (vérification de la validité des concepts), l'expérimentation scientifique et le design itératif.

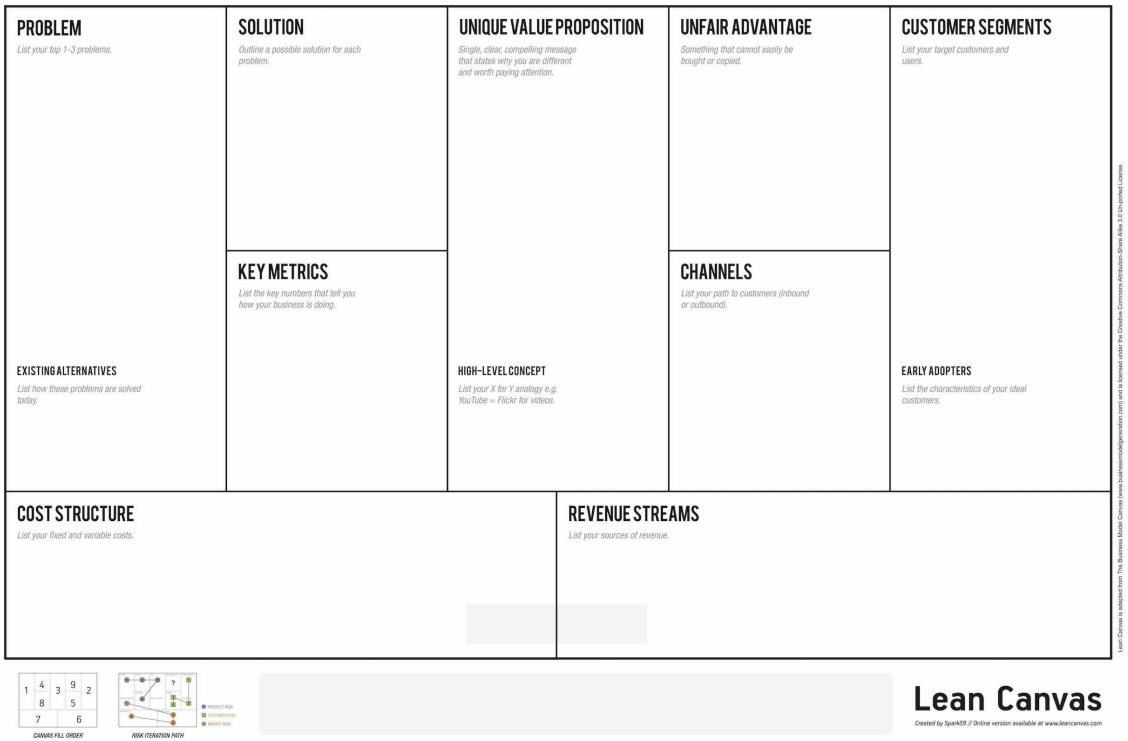
- cycles de commercialisation court des produits,
- mesurer régulièrement les progrès réalisés,
- obtenir des retours de la part des utilisateurs.

Lean Canvas : Cadrer un produit / service dans l’esprit agile

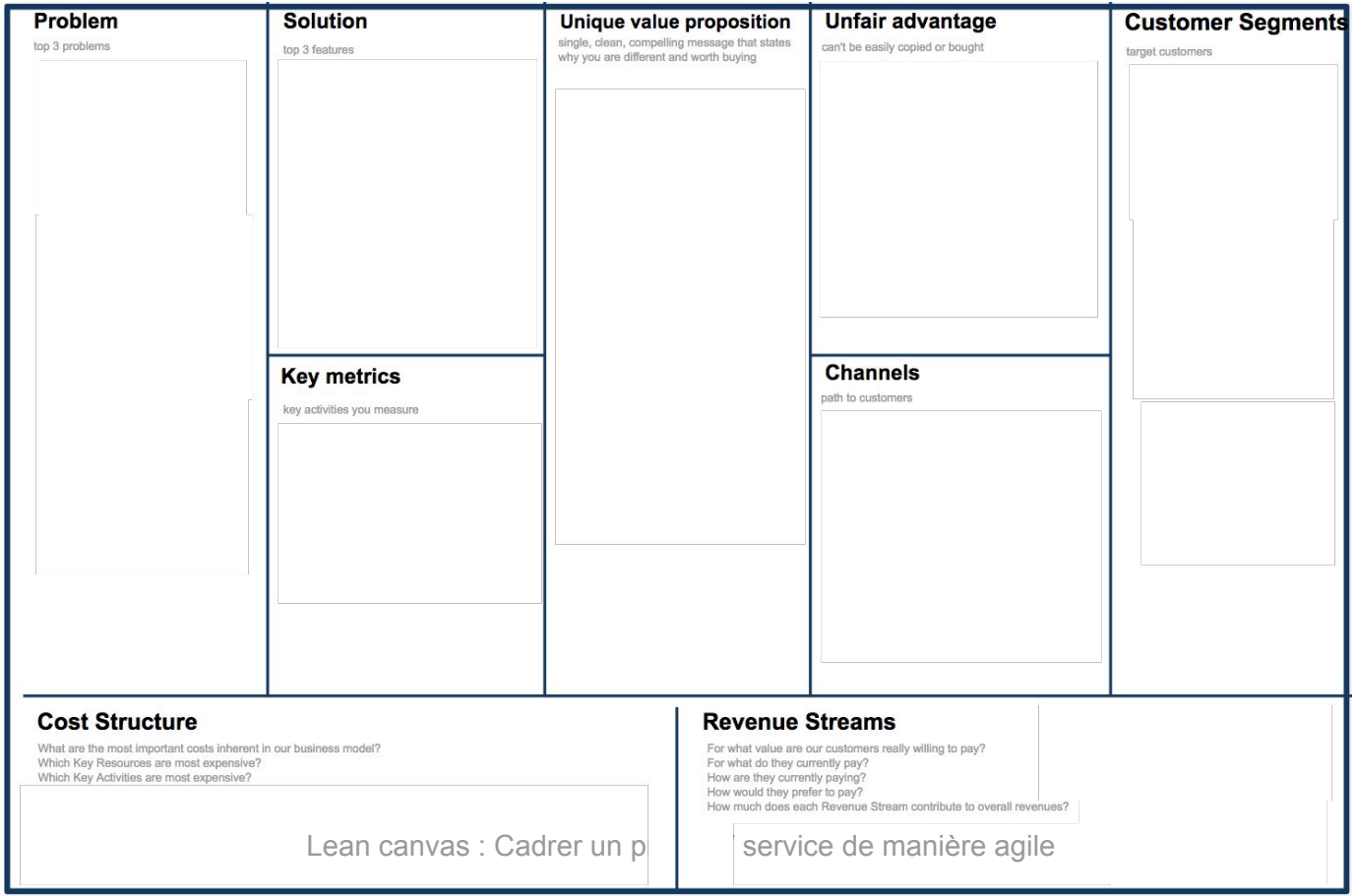
Lean Canvas : C’est quoi ?

LC : Confronter un produit à son marché

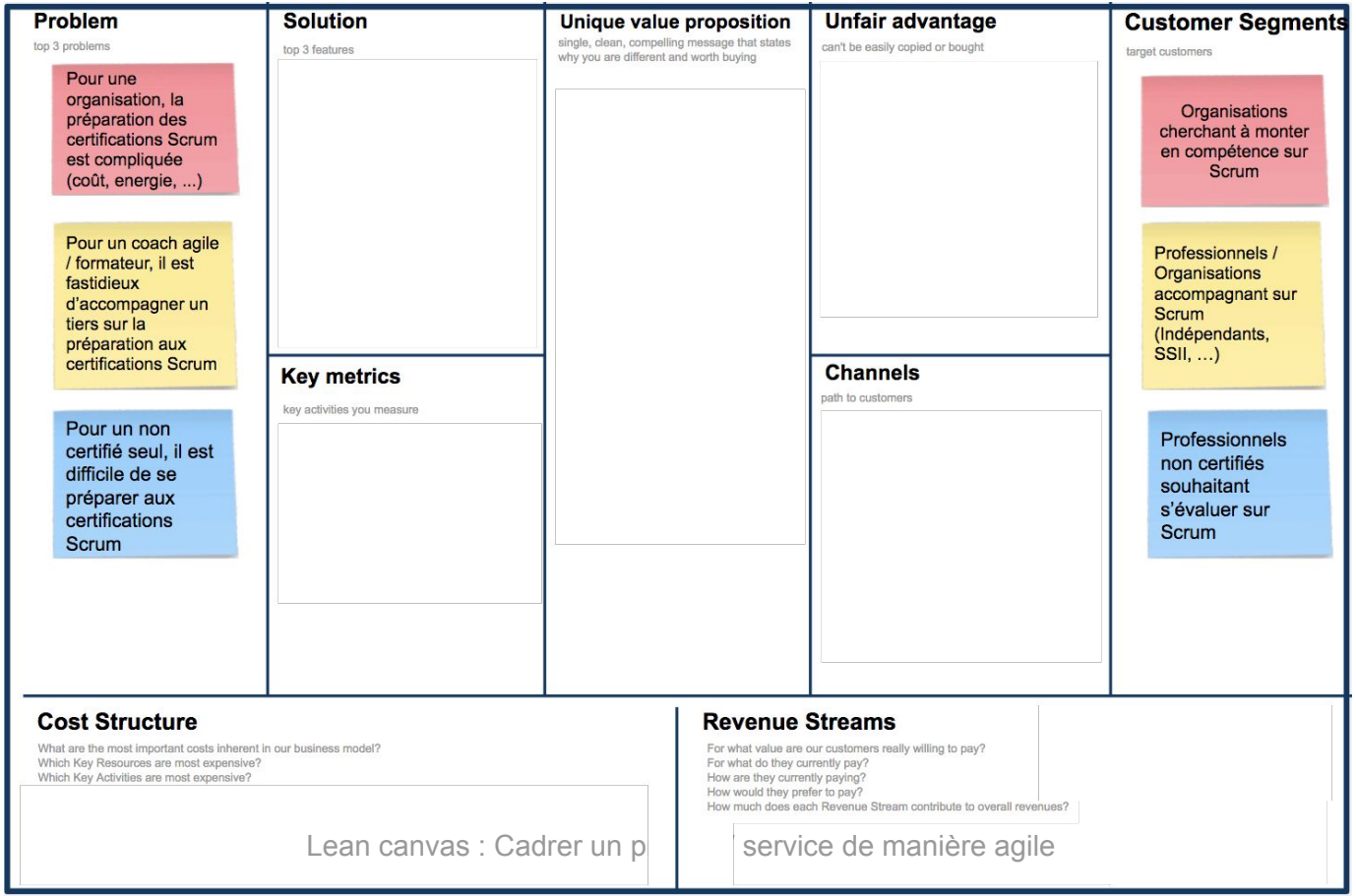
2014 : Ash Maurya reprend l’outil et l’adapte à la sauce Lean Startup.



Lean Canvas : Comment on fait ?

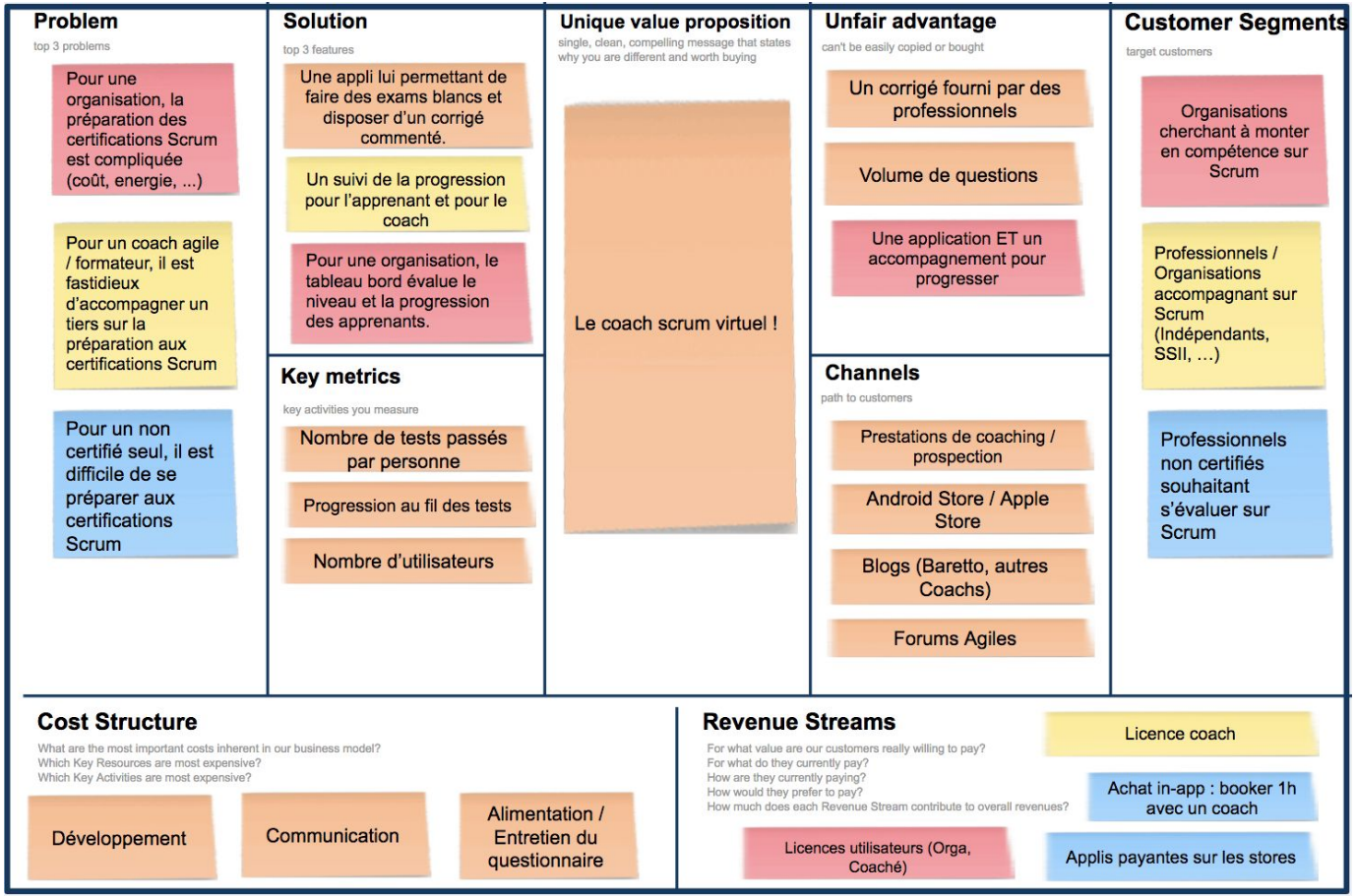


Lean Canvas : Comment on fait ?



Lean canvas : Cadrer un p service de manière agile

Lean Canvas : Comment on fait ?



XP

eXtreme programming

Ward Cunningham

A inventé et implémenté le concept de Wiki

Kent Beck

A lancé la série des X-Unit avec SUnit, en Smalltalk

Ron Jeffries

L'un des 17 signataires du manifeste agile

eXtreme programming

origines

origine :

Un projet chez Daimler Chrysler datant du milieu des années 90 et nommé le « Chrysler Comprehensive Compensation » ou « C3 »

Le logiciel d'origine contenait 2000 classes Smalltalk et 30 000 méthodes dont les modules transmettent des données faussent, bien que le programme réponde aux spécifications et cela malgré 18 mois de développement et des millions de dollars investis…

eXtreme programming

origines

origine :

Quand Kent Beck (un gourou reconnu du monde Smalltalk) repris le projet en 1996 accompagné de Ward Cunningham,

décide de repartir à zéro plutôt que continuer sur de mauvaises bases.

Ils compilent les pratiques de développement présentes sur les projets à succès, et les poussent à l'extrême Contrairement à d'autres méthodes qui s'occupent essentiellement de l'aspect organisationnel, l'eXtreme programming se penche principalement sur les pratiques techniques.

eXtreme programming

objectifs

Objectifs:

- Remettre le développeurs au centre de la production de logiciel
- Réduire les coûts du changement
- Améliorer le taux de succès dans les projets

Conçu en poussant à l'extrême les bonnes pratiques du développement:

- Revues de code -> revue permanente par du pair programming
- Conception -> refactoring tout au long du projet
- Tests unitaires -> Test Driven Development
- Intégration des développement -> Intégration continue, Déploiement continu

eXtreme programming

Valeurs

La communication :

- Intense et régulière
- Entre les clients/utilisateurs/développeurs/testeurs et décideurs

eXtreme programming

Valeurs

La simplicité :

- YAGNI (*you ain't gonna need it)* , KISS (*Keep it simple and stupid)*
- Ne pas chercher à anticiper les évolutions,Plus simple, plus rapide à livrer

eXtreme programming

Valeurs

Feedback :

- Echange entre le développeur et l'utilisateur
- Ecriture de tests unitaires, fonctionnels

eXtreme programming

Valeurs

Le courage :

- Accepter de tester, se tromper, refaire ..
- Ne pas cacher les difficultés

eXtreme programming

Valeurs

Le respect

- Du travail
- Des règles de l’équipe
- Des membres de l’équipe

Les pratiques

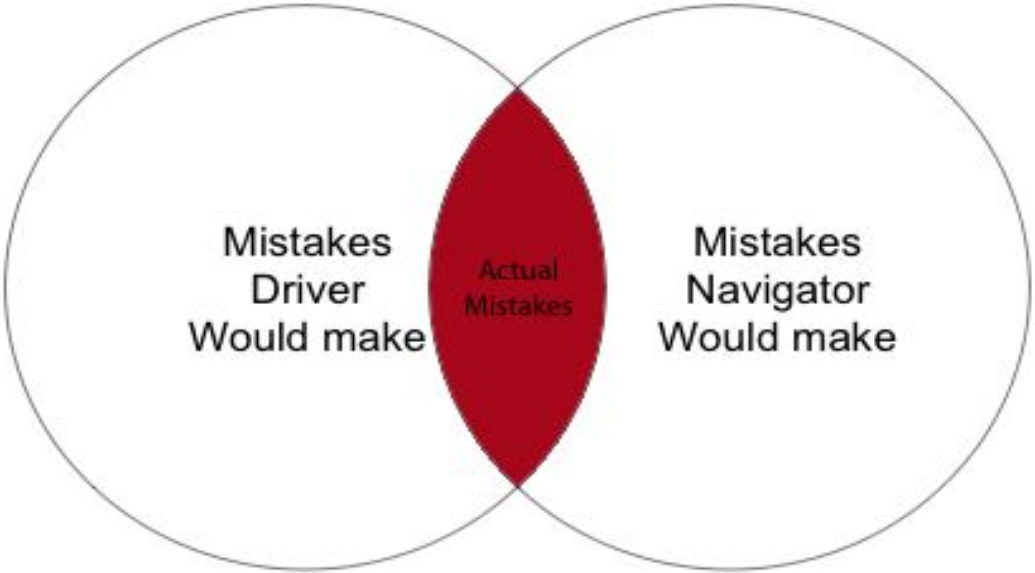
- Planning game
  - Planifier de manière incrémentale
- Small release
  - Petite livraison et régulièrement
- Metaphor
  - Ramener les idées complexes à des choses simples
- Simple design
  - KISS, YAGNI
- Testing
  - Tester au maximum, et automatiser
- Refactoring
  - Améliorer le design sans modifier le comportement
- Collective code ownership
  - Le code appartient à toute l’ équipe
- Continuous integration
  - Compilation régulière
- Sustainable pace
  - Intensité de travail soutenable
- Whole team
  - Membre plutôt généraliste
- Coding standard
  - Règles pour et par l’équipe
- On site customer
  - Représentant utilisateur sur site

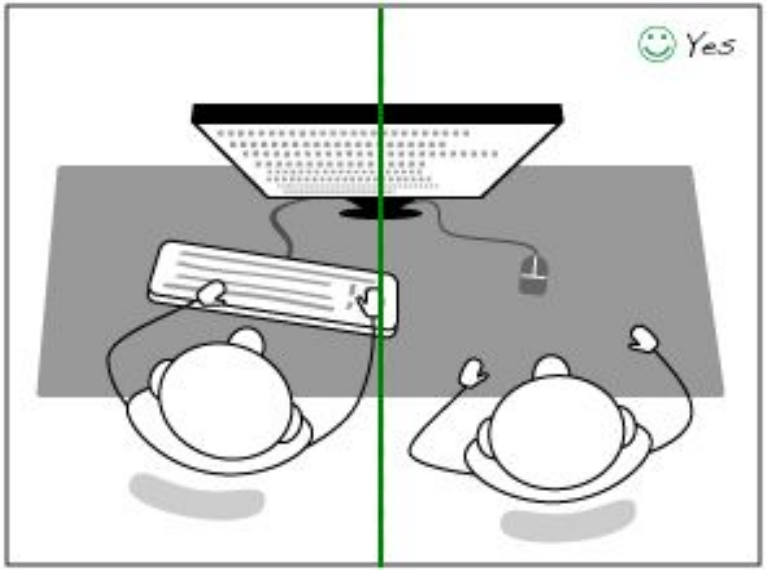
Pair programming

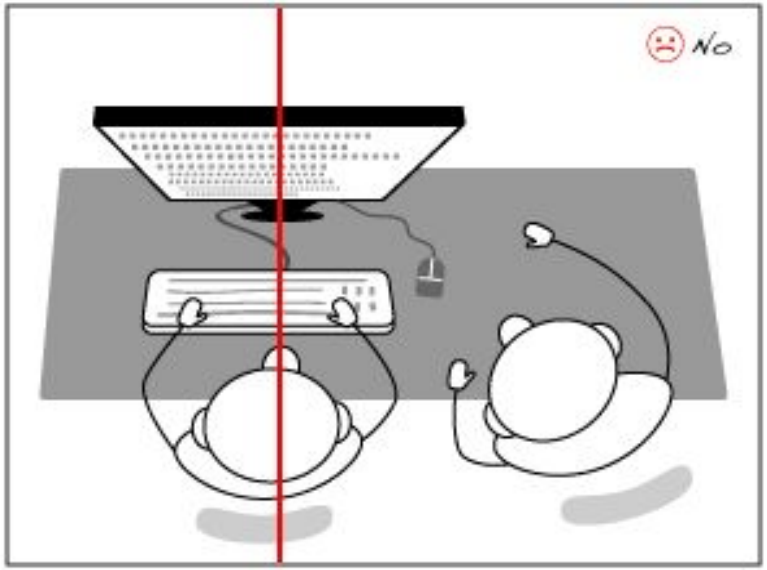
Ce qu’on cherche à atteindre en faisant du Pair Programming :

- Partager les connaissances
- Monter en compétence rapidement
- Casser l’isolation sur les périmètres (Ex : chacun son appli)
- Augmenter la qualité (revue en continue)

Pair programming




Yes

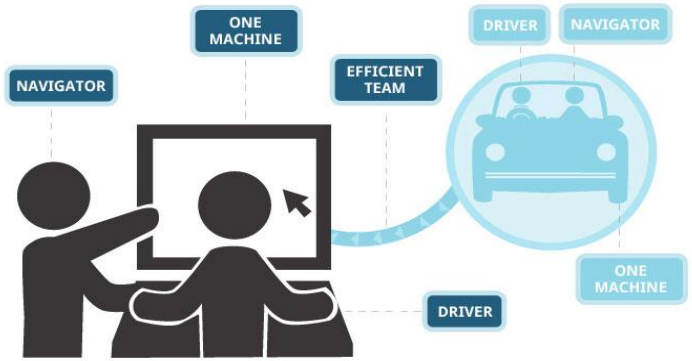

No

Pair programming

Pair Reviews

Pair learning

Pair debugging



Pair Pressure

Pair design

Pair Negociation

Pair programming

la limite c’est le cerveau

Un développeur passe plus de temps à lire du coder et à trouver quoi et où coder plutôt que de “taper” du code

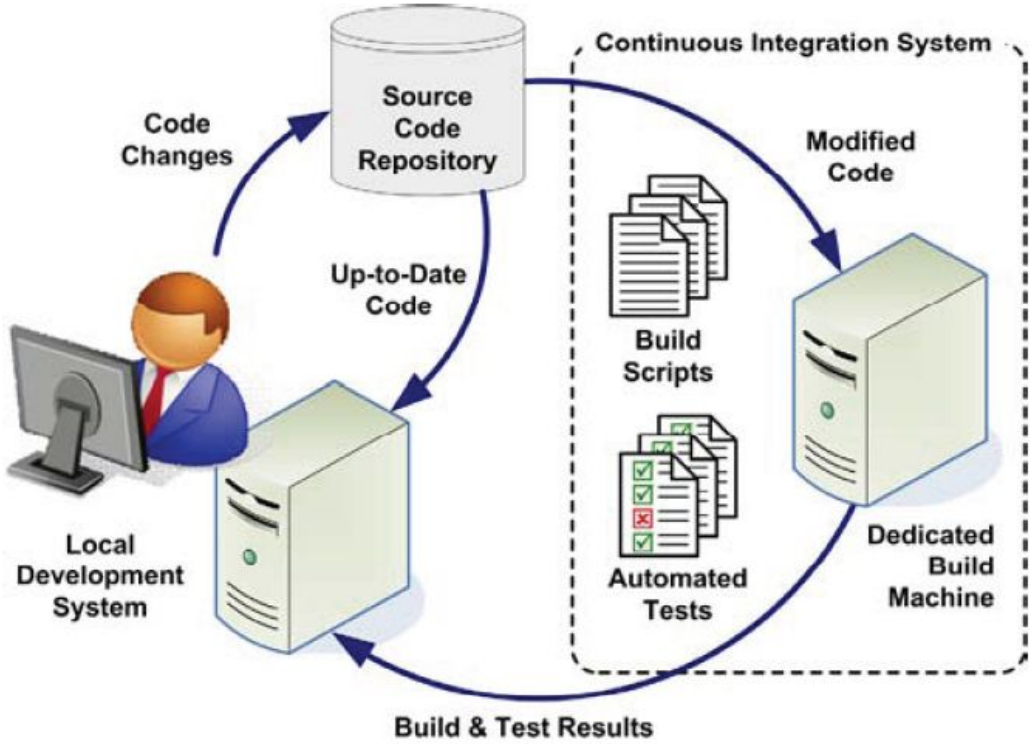
exemples :

- une application 100 000 lignes de code 8 Devs
- temps pour écrire une ligne code -> 10 secondes

→ temps pour réaliser une application de cette taille 1 ans

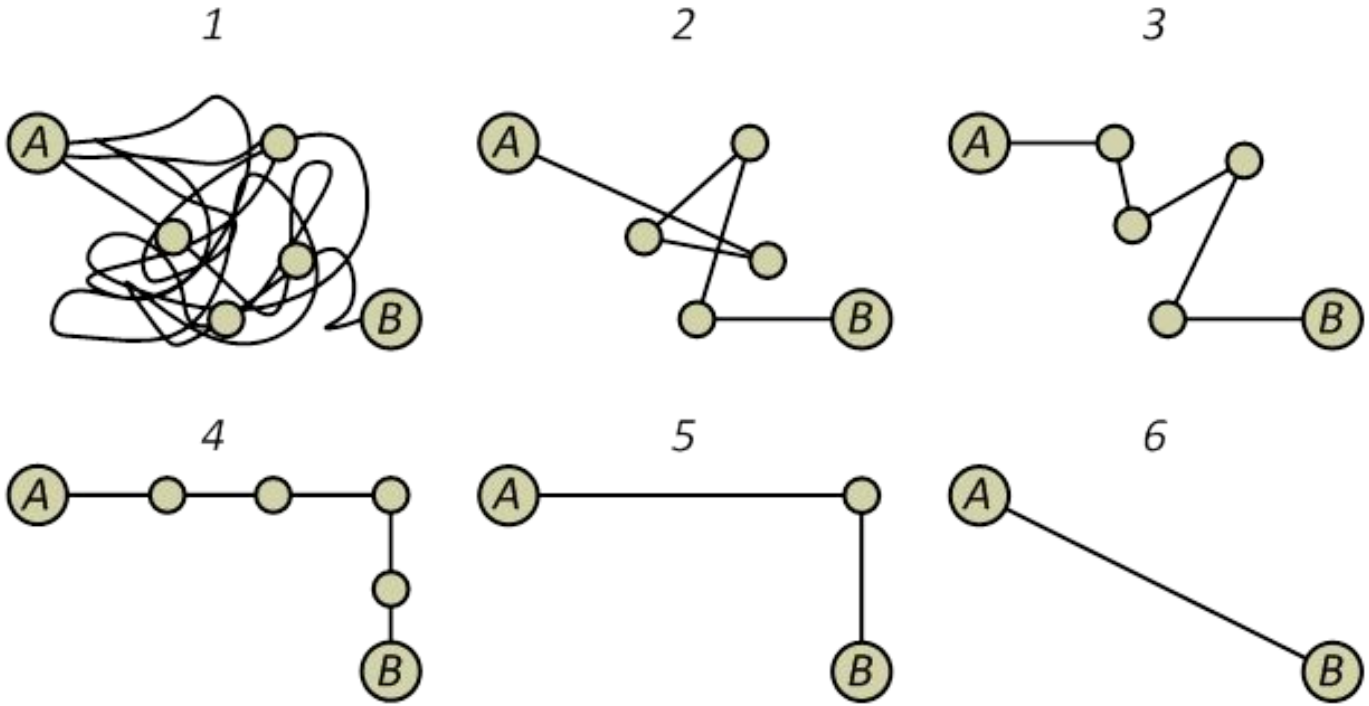
→ temps nécessaire pour écrire tout le code de l’application **5 jours**

Continuous integration

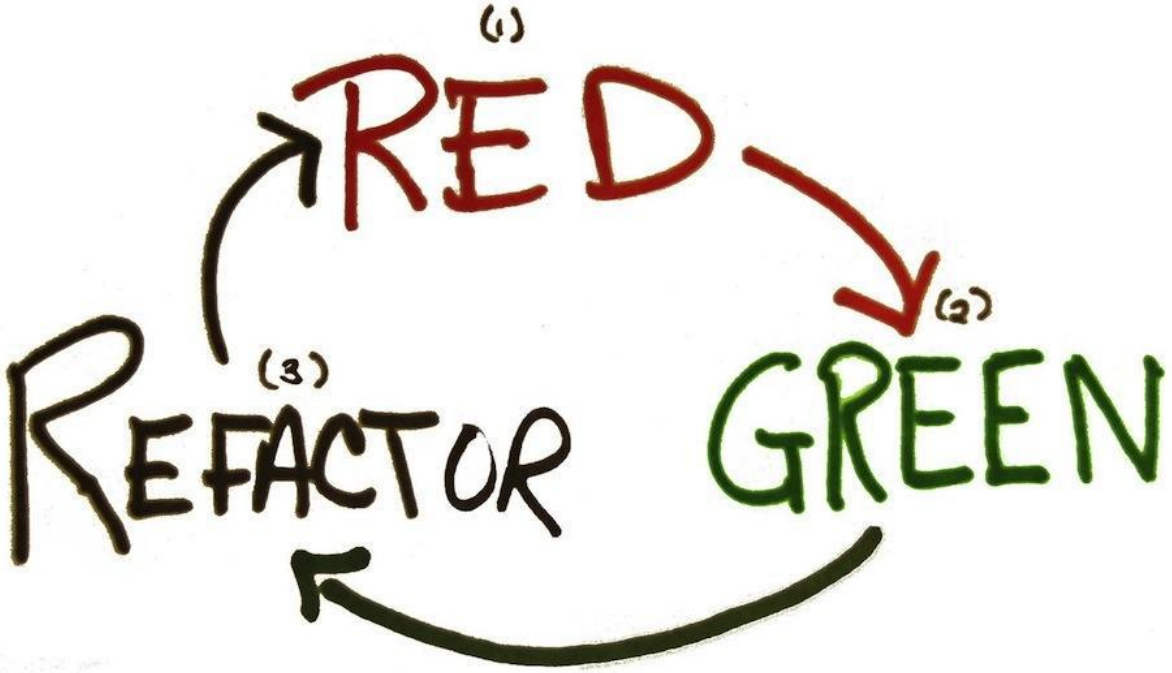


Refactoring

Procéder par petits pas...



TEST DRIVEN DEVELOPMENT



On écrit d'abord les tests, ensuite l'implémentation est réalisée.

- Grâce à cela, on s'assure d'avoir une fonctionnalité opérationnel,
- On peut refactorer sans crainte,
- On s'aperçoit des effets de bord au plus tôt (feedback)

User story

Un pattern venant d'XP

Introduit dans les années 2004-2005

Deux contributeur majeurs

Mike Cohn et Jeff Patton

Raconter une histoire...

Se décorréler de la technique et forcer l'empathie envers l'utilisateur

.... et discuter !

Mais surtout discuter pour développer et s'approprier l'histoire.
C'est un outil très utile pour s'assurer de la compréhension du message.

Un pattern simple à retenir...

En tant que … <utilisateur>
Je veux … <action>
Afin de … <cible/objectif>

… Mais difficile à maîtriser

Ne pas négliger l'utilisateur et toute la modélisation qui va avec (acteur, utilisateur, persona…).
Ne pas négliger l'objectif qui est un vecteur clé de l'agilité (adaptation au changement).

User story

Indépendante

Chaque story doit être livrable indépendamment des autres.

Negociable

La story doit rester suffisamment concise pour laisser place à la discussion.

porteuse de Valeur

Chaque story doit être formée de manière à apporter chacune de la valeur aux parties prenantes.

Estimable

Chaque story doit être estimable.

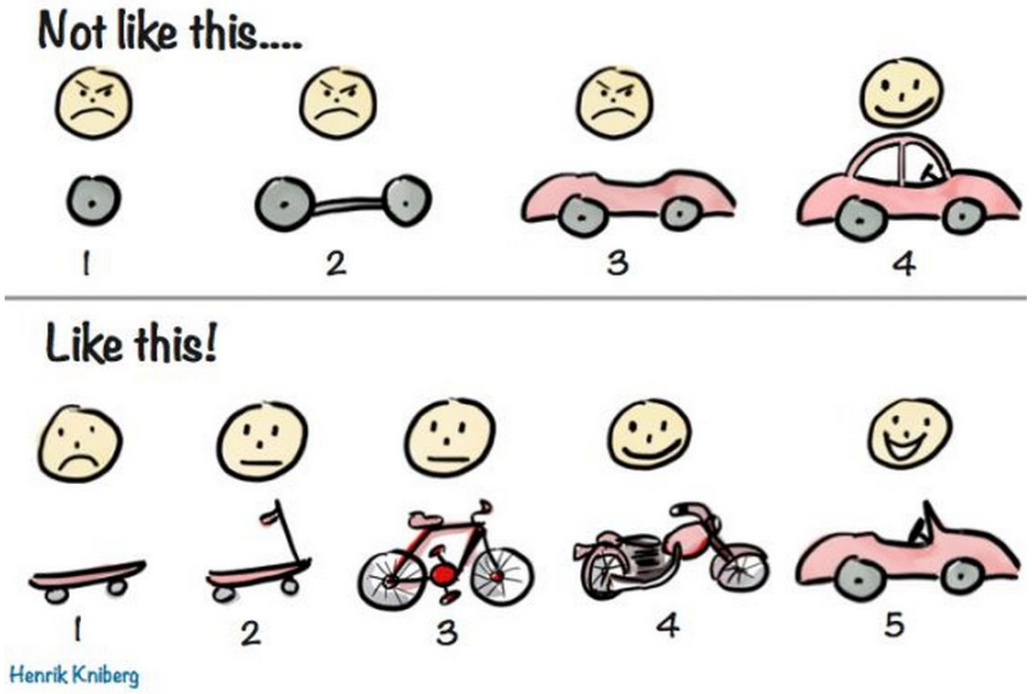
Suffisamment petite

La story doit pouvoir être traitée dans le cadre d'une itération.

Testable

La story ou sa description doit indiquer critères nécessaires à son test.

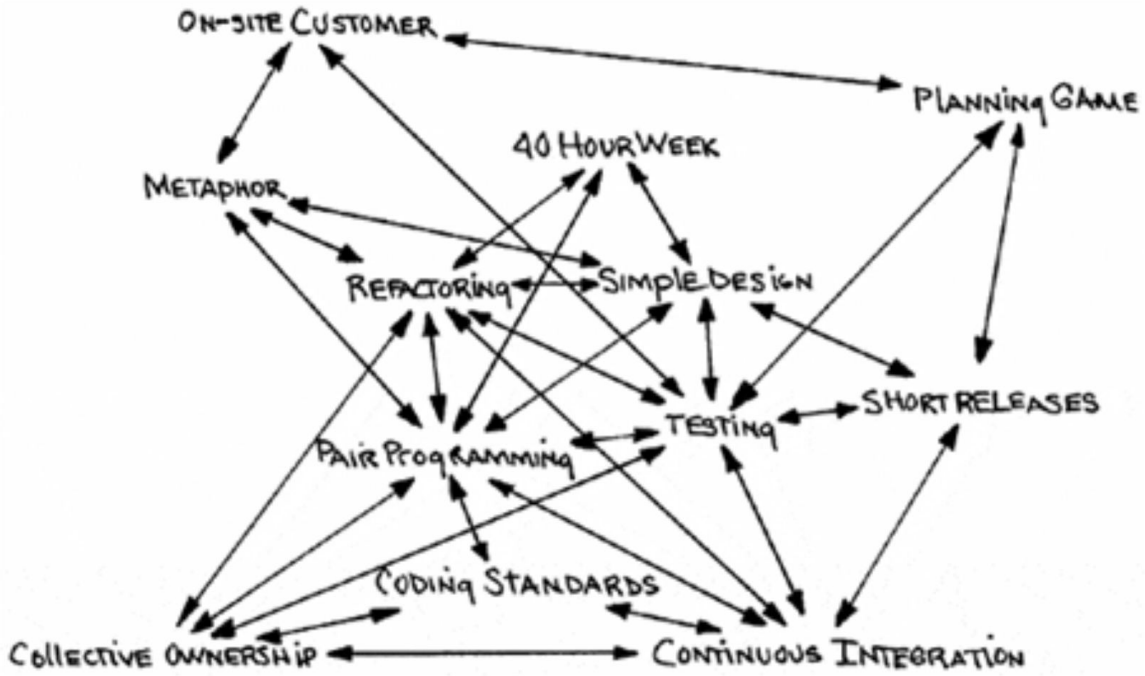
User story



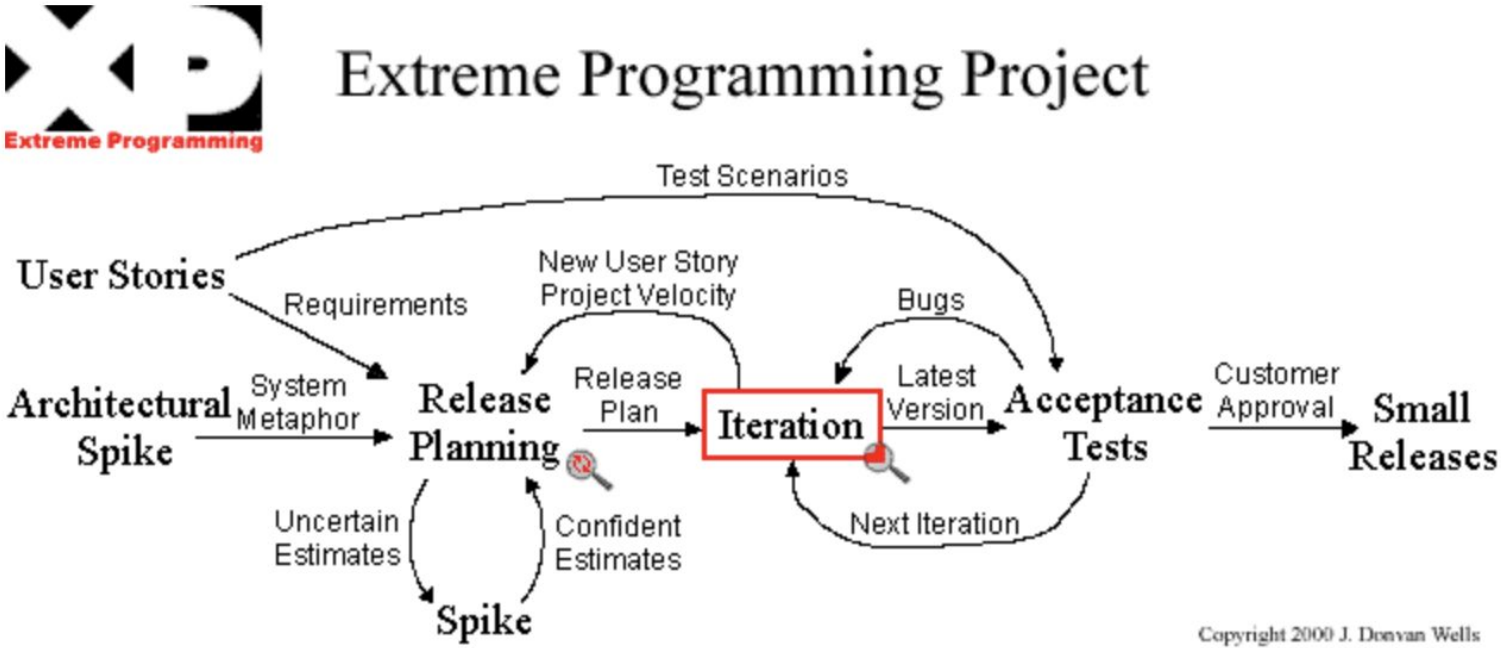
User story

	Stories	Use Cases	Requirements
Goal	generate conversation	capture a behavior	establish a contract
Scope	a single activity	a process "day in the life"	everything
Format	a single sentence	numbered bullets	specifications
Completeness	open for negotiation and refinements	locked, changes may impact entire process	locked, require scope change and approvals
Language	simple, comprehensible	structured, flows	precise, technical

Des pratiques qui fonctionnent en synergie



La recette magique : Itératif + Incrémental



https://quizizz.com/join?gc=225670

Scrum

Scrum

Ken Schwaber

L'un des fondateurs de l'Agile Alliance

Jeff Sutherland

Co-rédacteur du manifeste agile

Les piliers

Transparence :

- Les aspects importants du processus doivent être visibles.
- La transparence implique que soit défini un standard commun, une définition commune de « Fini » (Definition of Done).
- Scrum met l'accent sur le fait d'avoir un langage commun entre l'équipe et le management.

Les piliers

Inspection :

- Les utilisateurs de Scrum doivent régulièrement passer en revue les artéfacts et l'état d'avancement par rapport aux objectifs afin de détecter les écarts indésirables.

Les piliers

Adaptation :

Si une dérive est constatée pendant l'inspection, le processus / produit doit alors être adapté.
Scrum fournit des rituels, durant lesquels cette adaptation est possible.
Il s'agit de :

- La réunion de planification de sprint (sprint planning),
- La mêlée quotidienne (daily scrum),
- La revue de sprint (sprint review),
- La rétrospective de sprint (sprint retrospective).

Les artefacts

Backlog Produit

Le Backlog Produit est une liste ordonnée de tous les éléments identifiés comme nécessaires au produit. Il constitue l’unique source d'exigences pour tout changement à apporter au produit. Le Product Owner est responsable du Backlog produit, y compris son contenu, sa disponibilité et son ordonnancement.

Les artefacts

Backlog Sprint

Le Backlog Sprint est l’ensemble des éléments sélectionnés pour le Sprint plus un plan pour livrer l’incrément du produit et réaliser l’objectif du Sprint. Le Backlog Sprint est une prévision que l’équipe de développement fait de la fonctionnalité qui sera présente dans le prochain incrément et le travail nécessaire pour livrer cette fonctionnalité dans un incrément « Fini ». Le Backlog Sprint rend visible tout le travail que l'équipe de développement identifié comme nécessaire pour atteindre l'objectif du Sprint.

Les cérémonies

Le Sprint

Le cœur de Scrum est le Sprint, qui a une boîte de temps (time-box), une durée, d'un mois ou moins au cours de laquelle un Incrément Produit « Fini » fonctionnel et potentiellement publiable est créé. Les sprints ont une durée cohérente durant la phase de développement. Un nouveau Sprint commence immédiatement après la conclusion du Sprint précédent.

Les cérémonies

Daily Scrum

La mêlée quotidienne (Daily Scrum) est un événement de 15 minutes (time-boxé) destiné à l'équipe de développement.

La mêlée quotidienne est tenue tous les jours du Sprint. L'équipe de développement planifie le travail pour les prochaines 24 heures. Cela optimise la collaboration et la performance de l'équipe tout en inspectant le travail depuis la dernière mêlée quotidienne et envisageant le travail restant durant le Sprint.

La mêlée quotidienne est tenue à la même heure et au même lieu chaque jour pour réduire la complexité.

Les cérémonies

Revue de Sprint

Une revue de Sprint (Sprint Review) est tenue à la fin du Sprint pour inspecter l’incrément réalisé et adapter le Backlog Produit si nécessaire.

Pendant la revue de Sprint, l'équipe Scrum et les parties prenantes échangent sur ce qui a été fait durant le Sprint.

Les cérémonies

Rétrospective de Sprint

La rétrospective de Sprint (Sprint Retrospective) est une opportunité pour l'équipe Scrum de s’auto-inspecter et de créer un plan d'améliorations à adopter au cours du prochain Sprint.

Les cérémonies

Product Backlog Refinement

Le raffinement du backlog de produit (Product Backlog Refinement) consiste en l'ajout de détails, d'estimations et de l'ordonnancement des éléments du Backlog Produit. Il s'agit d'une activité régulière dans laquelle le Product Owner et l' équipe de développement collaborent pour détailler les éléments du Backlog Produit. Durant le raffinement du Backlog Produit, les éléments sont revisités et révisés.

Les rôles

Product Owner : PO

Development Team : Dev team

Scrum Master : SM

Le rôle de PO dans Scrum

"By the book"

Maximiser la valeur du produit et de ce qui sera fait par la DevTeam

Cela se fait au travers de la gestion du product backlog, dont le PO est le **seul responsable**.

Que comprend la gestion du product backlog ?

Exprimer clairement les items.

Prioriser les items de la meilleure manière pour atteindre les objectifs et missions.

S'assurer que le backlog est **visible, transparent et clair pour tous** et qu'il montre ce sur lequel l'équipe travaillera ensuite.

S'assurer que la DevTeam a un niveau de compréhension suffisant des items du backlog.

Le rôle de PO dans Scrum

Quelques compléments

Le PO est responsable du “Quoi”, pas du “Comment”

C’est la DevTeam qui est pleinement compétente pour assumer le “comment”.

Par conséquent, en tant que PO, il ne faut pas…

Estimer les items à la place de la devTeam .

Décider de la conception technique / du découpage en tâches à la place de la devTeam.

Décider de l’ordre dans lequel les tâches du sprint seront réalisées (elles appartiennent au **Sprint Backlog**).

Manager l’équipe (elle est auto-organisée !).

Le rôle de Scrum Master

"By the book"

- Il comprend complètement l'agile et Scrum
- Il est garant de l'agilité sur le projet, coache la Dev. Team et s'assure de la bonne mise en œuvre de l'agilité par la Dev. Team
- C'est un rôle de management : il manage le process Scrum. Il ne manage pas la Dev team (il n'est donc pas hiérarchiquement supérieur).
- Il veille au bon déroulement du projet et contribue à enlever les freins et obstacles qui empêchent la Dev Team d'aller à sa vitesse optimale

Le rôle de Scrum Master

"By the book"

- Il aide aussi le Product Owner dans son activité
- Il isole/ protège l'équipe des perturbations extérieures
- Il apprend à ce que la Dev team effectue les cérémonies dans la forme adéquat (time-box…) - Animateur des cérémonies agiles – à minima fait en sorte qu'elles aient lieu et se passent correctement

La Dev Team

"By the book"

L'équipe de développement se compose de professionnels qui fournissent un incrément « Fini » potentiellement publiable (Releasable) à la fin de chaque Sprint. Un incrément « Fini » est requis à la revue de sprint. Seuls les membres de l'équipe de développement créent l'incrément.

Les équipes de développement sont structurées et habilitées par l'organisation à s'organiser et gérer leur propre travail. La synergie résultante optimise l'efficience et l'efficacité globale de l'équipe de développement.

La Dev Team

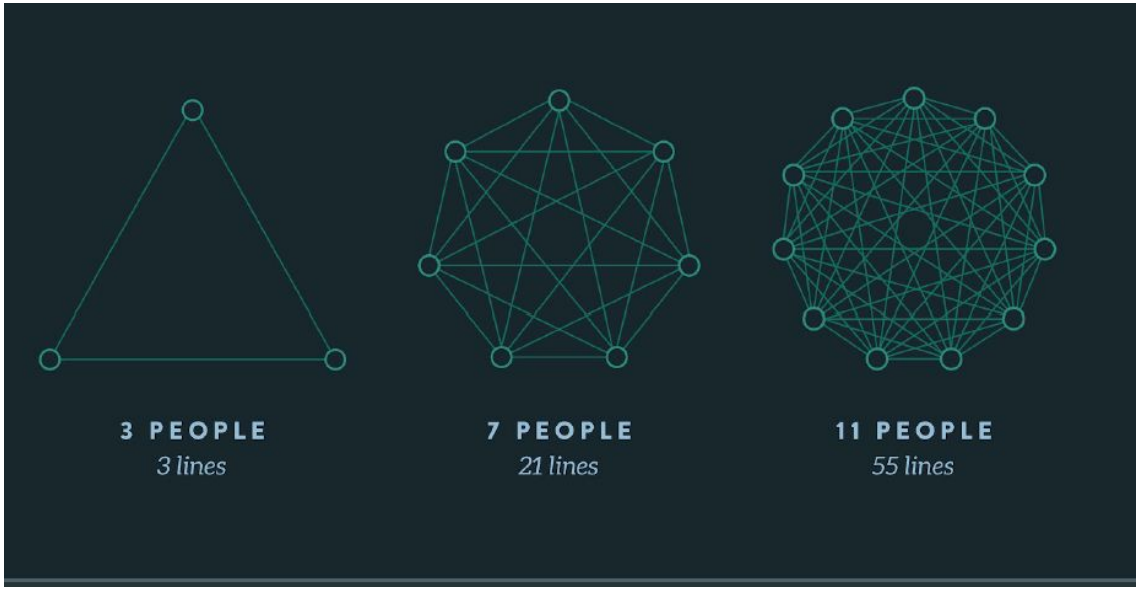
"By the book"

- Elle se partage la responsabilité de la partie technique du projet
- Cross fonctionnelle & Pluridisciplinaires
- Auto-organisée
- Effectue les estimations
- Sollicite le Product Owner dès que nécessaire
- Résout ses propres conflits
- Pas de hiérarchie , pas de structure , pas de rôle prédéfinis
- Chaque membre peut potentiellement être amené à effectué toutes les activités de l'équipe

La Dev Team

"By the book"

La taille optimale de l'équipe de développement doit être suffisamment petite pour rester réactive et assez grande pour accomplir un travail significatif durant le Sprint (3 à 9)

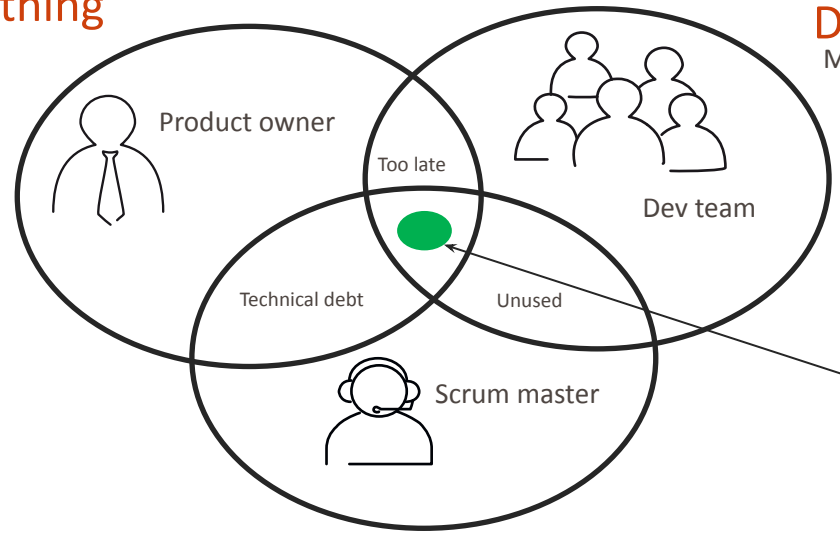


La Scrum Team

Learn, improve, innovate by enabling a fast feedback loop

Do the right thing
Make things lovable

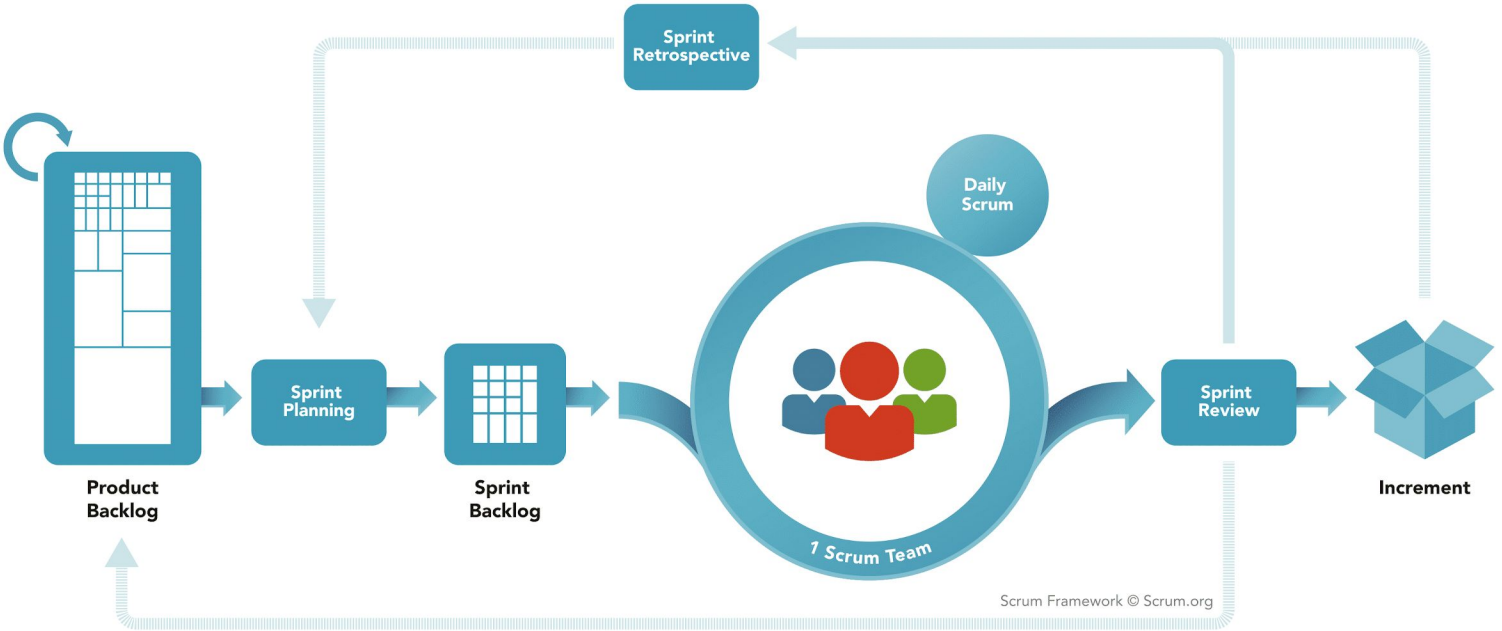
Do the thing right
Make things maintainable



Lead teams there !!!!!
The MAGIC ZONE

Do the thing fast

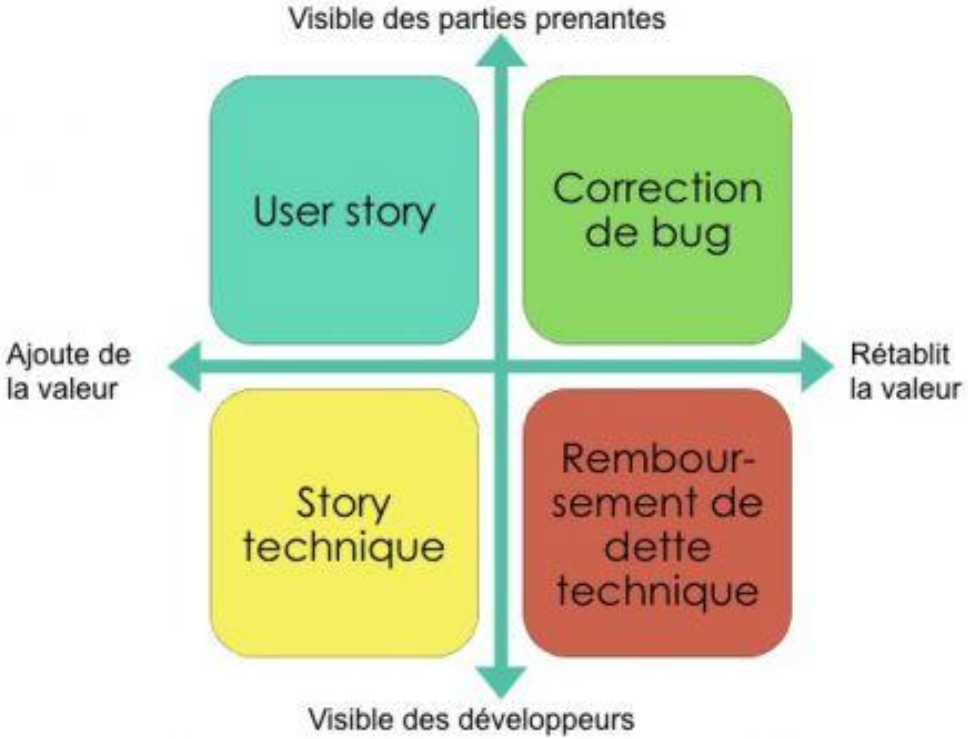
SCRUM FRAMEWORK



Scrum.org

Composition du product backlog

Types d'items

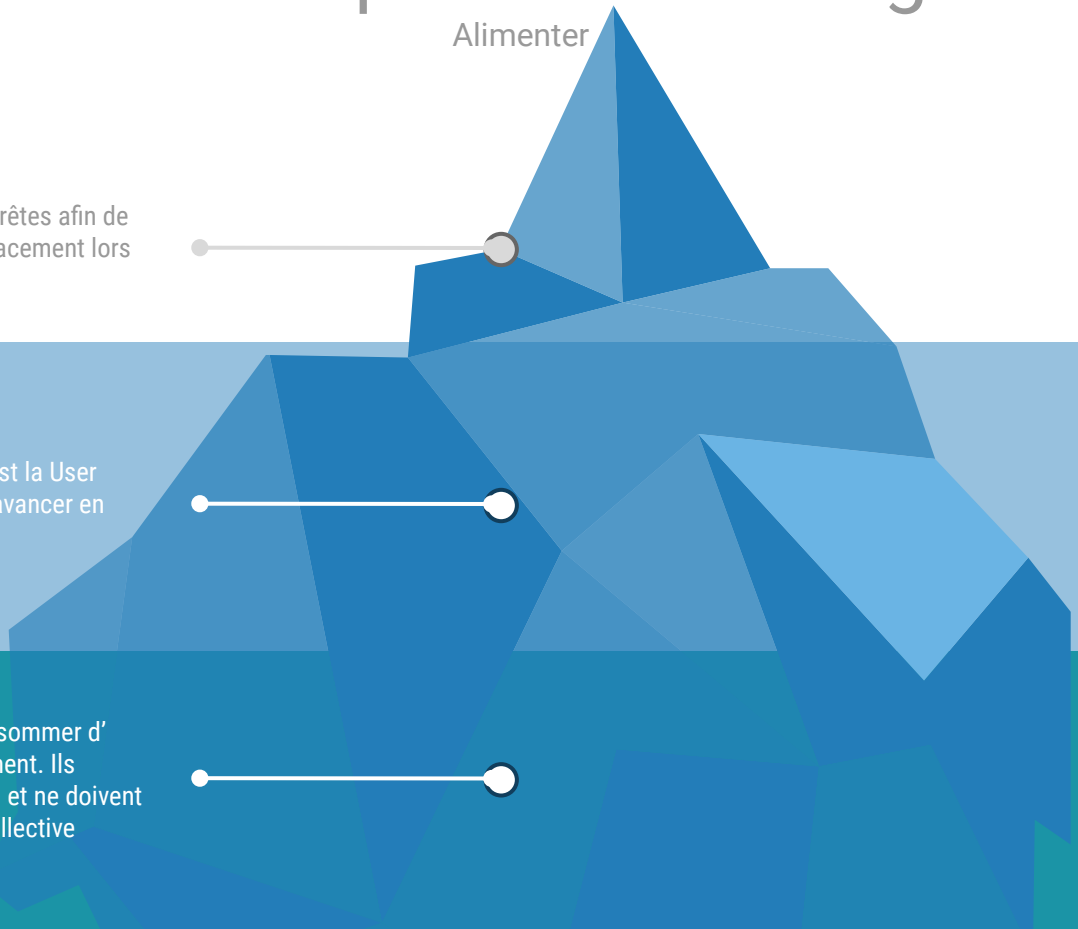


Comment constituer le backlog ?

- **Ateliers Utilisateurs**
- **Observation des Utilisateurs**
- **Les Statistiques d’utilisation**
- **L’analyse du marché**
- **La vision du PO et de l’équipe**

Vie du product backlog

Alimenter



Les user stories doivent être prêtes afin de pouvoir être embarquées efficacement lors du planning de sprint.

Sprint +1

La granularité la plus grosse est la User Story. Ces sujets sont ceux à avancer en backlog refinement.

Version +1

Ces sujets ne doivent pas consommer d'énergie à l'équipe pour le moment. Ils constituent le fond du backlog et ne doivent pas faire l'objet d'une revue collective poussée.

Au delà

Vie du product backlog

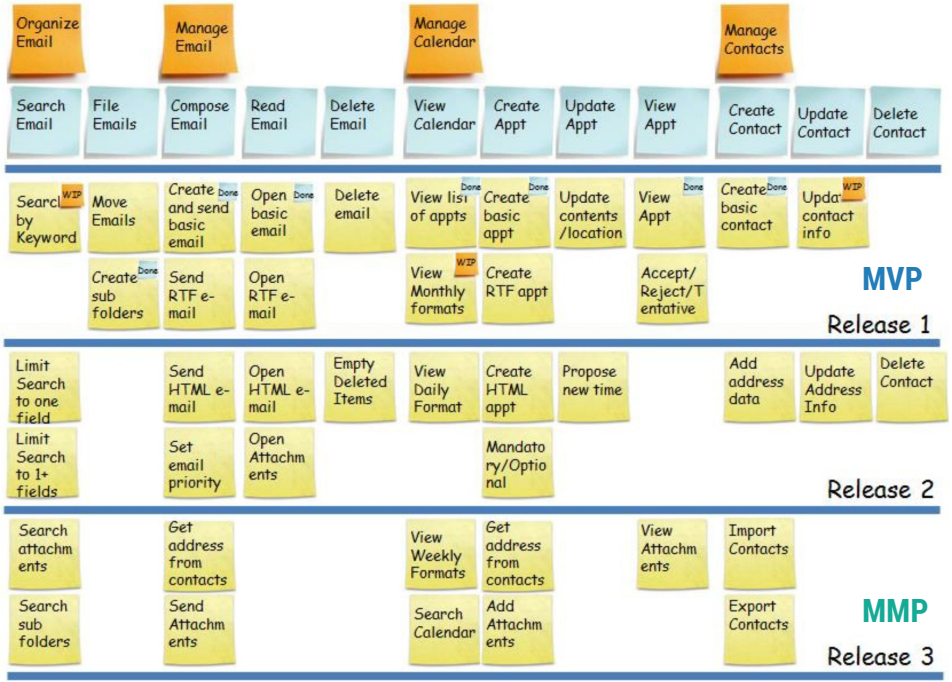
Prioriser : MVP / MMP

Minimum Viable Product (MVP)

L'ensemble minimal de fonctionnalités permettant de rendre un service / portant une valeur ajoutée. Il est plutôt destiné à des utilisateurs pilotes / volontaires.

Minimum Marketable product (MMP)

L'ensemble minimal de fonctionnalités permettant de lancer le produit sur le marché / d'être vendu. Il est destiné à n'importe quel utilisateur potentiel.



https://quizizz.com/join?gc=009539

Conclusion

Conclusion

- Pas de solution miracle
- D’abord comprendre les principes des méthodes et des pratiques.
- Adapter les méthodes en cas de besoin
- Savoir ce que les pratiques apportent