Architecture Logicielle Agile v.2

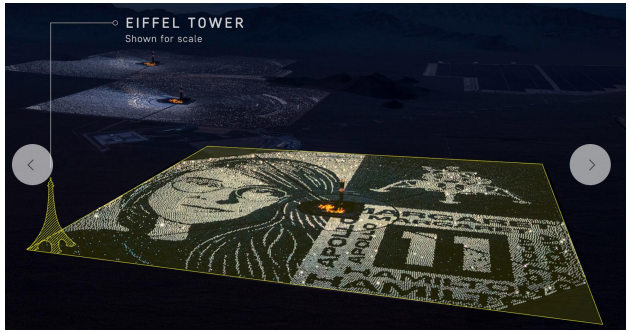
Sandy Ingram

Spring 2024 – Micro 371

Plan du cours

- L'ingénierie Logicielle:
  - Définition (avec un peu d'histoire)
  - Le DevOps
  - L'agilité
- Approche de développement "Waterfall"
- Ingénierie et Architecture Logiciel Agile
  - Origine
  - Définition
  - "Agile Manifesto": définition et principes
- Exemples de méthodologies agile:
  - XP pour extreme programming
  - Kanban
  - (Focus sur) SCRUM
- Synthèse et limitations
- Méthodologie agile dans la pratique:
  - Planification agile avec outils gitlab.

L'ingénieurie Logicielle

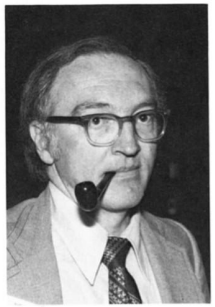


A Google tribute to Margaret Hamilton (Software Team Lead for Apollo mission flights.)

"When I first came up with the term, no one had heard of it before, at least in our world. It was an ongoing joke for a long time. They liked to kid me about my radical ideas. It was a memorable day when one of the most respected hardware gurus explained to everyone in a meeting that he agreed with me that the process of building software should also be considered an engineering discipline, just like with hardware. Not because of his acceptance of the new 'term' per se, but because we had earned his and the acceptance of the others in the room as being in an engineering field in its own right" (Margaret Hamilton).

L'Ingénieurie Logicielle

Première conférences : 1968 et 1969 sponsorisés par le comité scientifique de l'OTAN. Personnes clés:



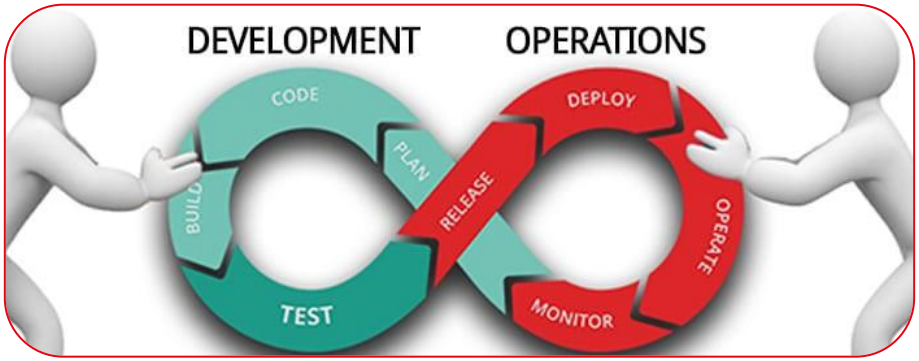
Douglas Ross
(Organisateur)

Anthony Oettinger: aussi crédité pour la première utilisation de "Software Engineering" en 1968.

1990s: première utilisation du mot "architecture" dans un livre de développement logiciel.

L'ingénieurie Logicielle : 2 pilliers

DevOps

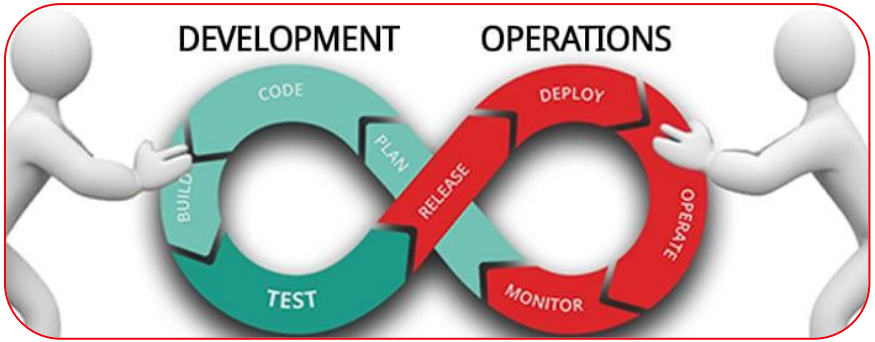


- *Selon le rapport de 2013 "State of DevOps" de Puppet Labs, les entreprises pratiquant le DevOps déploient 30 fois que leurs compétiteurs et moins de 50% de leurs déploiements échouent.*

Agilité (Le focus de ce cours)

L'ingénieurie Logicielle : 2 pilliers

DevOps



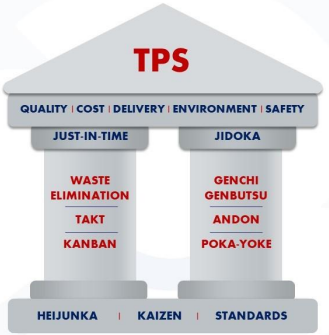
Selon le rapport de 2013 "State of DevOps" de Puppet Labs, les entreprises pratiquant le DevOps déploient 30 fois que leurs compétiteurs et moins de 50% de leurs déploiements échouent.

Agilité

Ce que nous traiterons aujourd'hui

A l'origine :
Le Lean manufactoring

- Introduit dans les années 80s, le terme "Lean" est mis en avant dans les années 90s, avec le livre "The Machine That Changed the World" pour décrire l'approche efficiente et "anti-gaspillage" de Toyota Production Systems (TPS) se caractérisant par:
  - un flux de production **continu**
  - **amélioration** continue (Kaizen)
  - un processus de qualité intégré
  - L'implication active des **employés**
  - Une production "JIT" ("just-in-time")
  - La flexibilité et **adaptabilité** au changement
  - Une minimisation des stocks et des **délais** de production



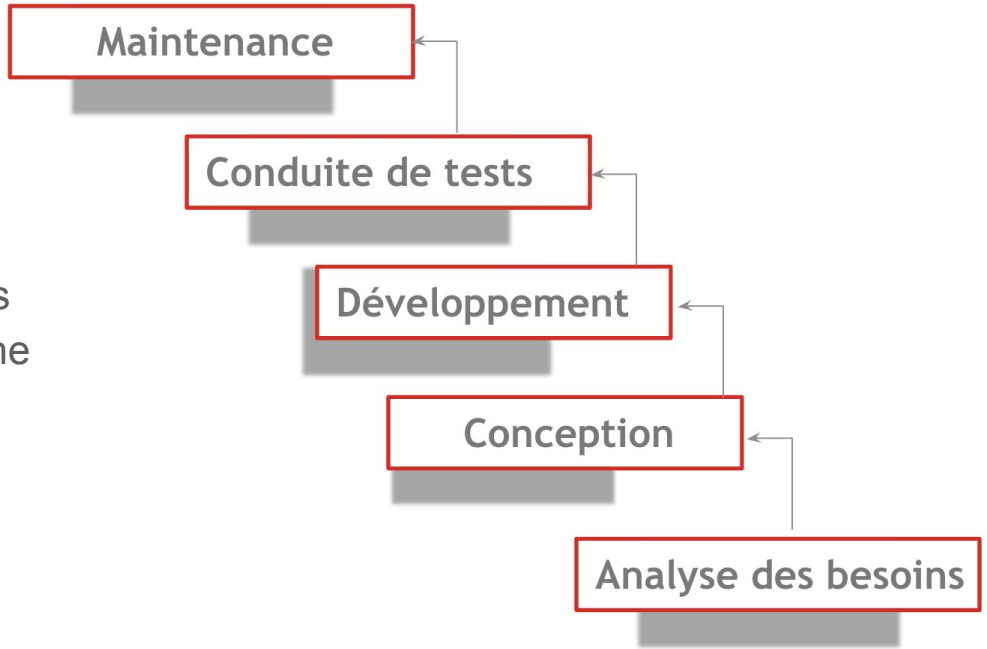
https://www.linkedin.com/pulse/tps-lean-fundamentals-special-offer-kaizen-made-easy/?trk=public_post_main-feed-card_feed-article-content

Approche de développement Agile

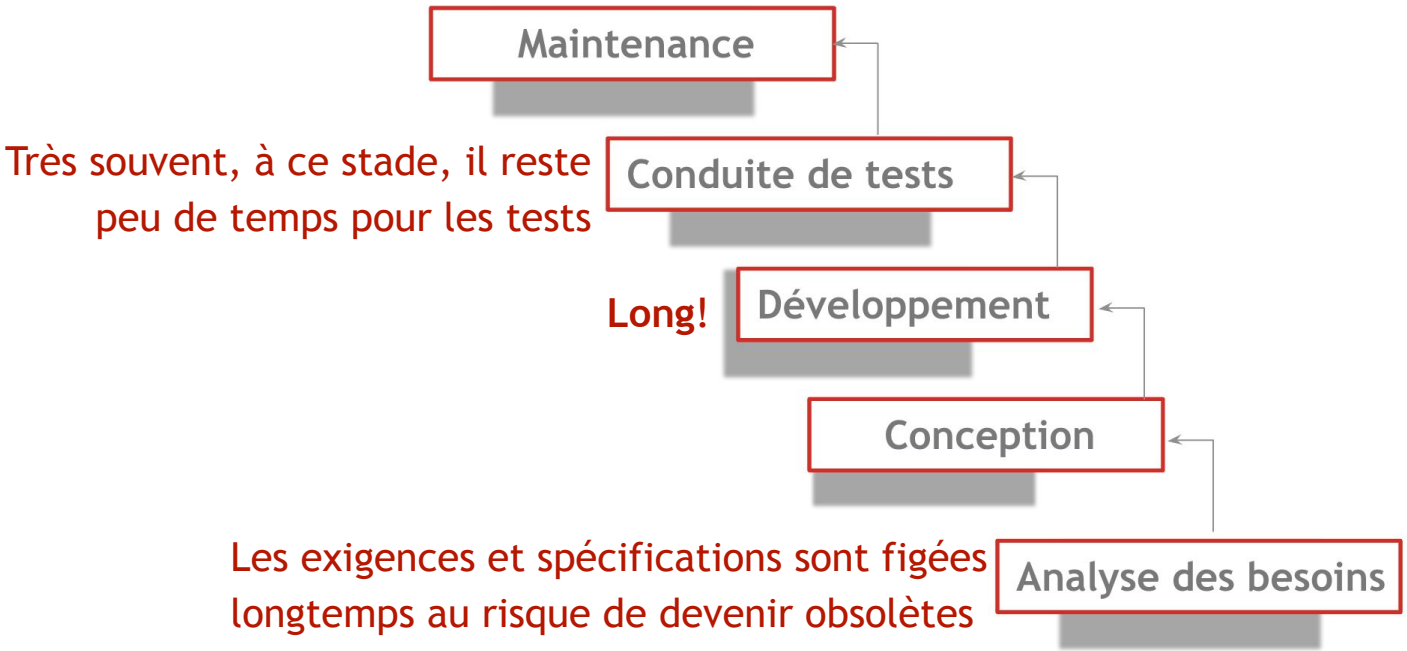
- Ken Schwaber and Jeff Sutherland proposent le “framework” SCRUM, publié en **1995**.
- Après avoir adopté le modèle “waterfall”, the Département de défense aux USAs dans son standard (DOD- STD-2167) décrit les processus itératifs dans les années 1990s.
- 17 “practiciens” de développement de logiciel réunis en **2001** pour discuter des meilleures pratiques de développement logiciels, publient l’Agile Manifesto. Donc SCRUM, XP, ont précédé à ce document qui en a fait un bilan.

Le problème avec le modèle “WaterFall”

La conception puis la réalisation du projet, sont réalisées de manière à tous les aspects du projet, en une fois; le projet est abordé dans sa **totalité**.



Le problème avec le modèle “WaterFall”



Sandy Ingram

Approches Agiles

Sandy Ingram

Lightweight approaches	Fuller approaches (beyond 1 team)
Scrum	Scrum-of-Scrums
Lean software development	Scrum at Scale (**Scrum@Scale**)
Kanban (process + method)	Large-scale Scrum (**LeSS**)
Extreme Programming (**XP**)	Scaled Agile Framework (**SAFe**)
Continuous Integration (**CI**)	Disciplined Agile Delivery (**DAD**)
Continuous Delivery (**CD**)	Dynamic Systems Development Method (**DSDM**)
Feature Driven development (**FDD**)	Agile Project Management (**AgilePM**)
Test Driven Development (**TDD**)	Agile Unified Process (**AUP**)
Crystal Clear	Open Unified Process (**OpenUP**)
...	...

http://www.nmerge.com/agile-marketing-lightweight-or-a-fuller-approach/agile/

“Agile Manifesto” 4 valeurs fondamentales

Sandy Ingram

“Agile Manifesto” 12 principes

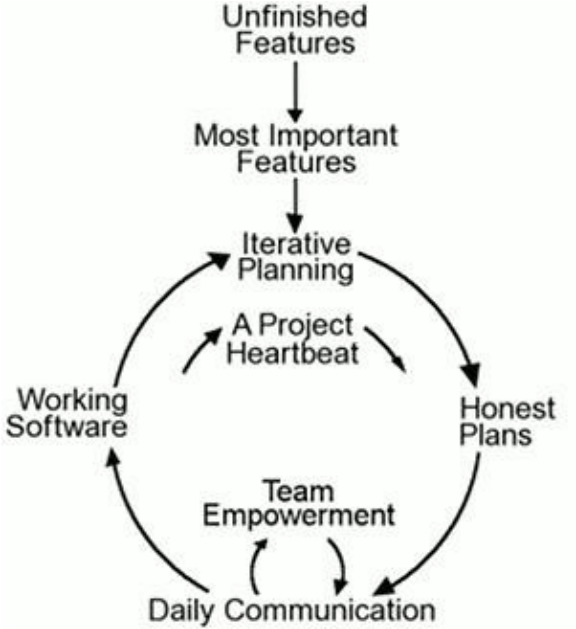
La **priorité** absolue est la satisfaction du client grâce à la livraison rapide et continue de logiciels fonctionnels et utiles.	Coopération **quotidienne** entre développeurs et porteurs du projet (business)
Intégration précoce et continue, publications fréquentes*	**Rythme** de développement soutenable
Attention permanente à l'excellence technique (bonne conception de l'architecture, automatisation des tâches répétitives, minimisation de la "dette" technique).	**Simplicité**
Un **logiciel fonctionnel** est le premier critère de réussite.	**Acceptation** des changements des requirements “même tardifs”
Construire des projets autour de **personnes** motivées	Des équipes “autonomes” au niveau de leur organisation
Rien n'équivaut à une conversation **en face à face.**	**Rétrospectives** et adaptation régulières (du processus)

* Pour le permettre viser l’automatisation de tests, l’intégration et le déploiement continu (CI/CD)

Sandy Ingram

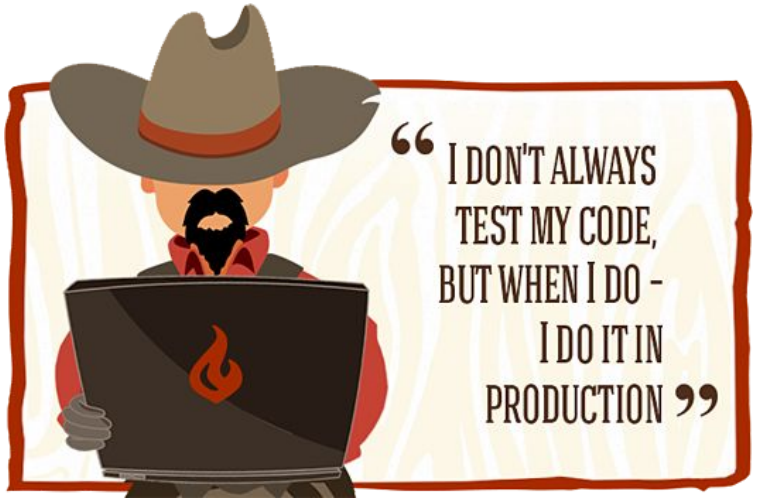
La Méthodologie Agile

Sandy Ingram



https://medium.com/@maruvada.krishna/product-development-using-agile-methodology-81b39c29f250

Agile n'est pas équivalent à CowBoy Coding!



https://www.cs.utexas.edu/blog/cowboy-rides-away-now

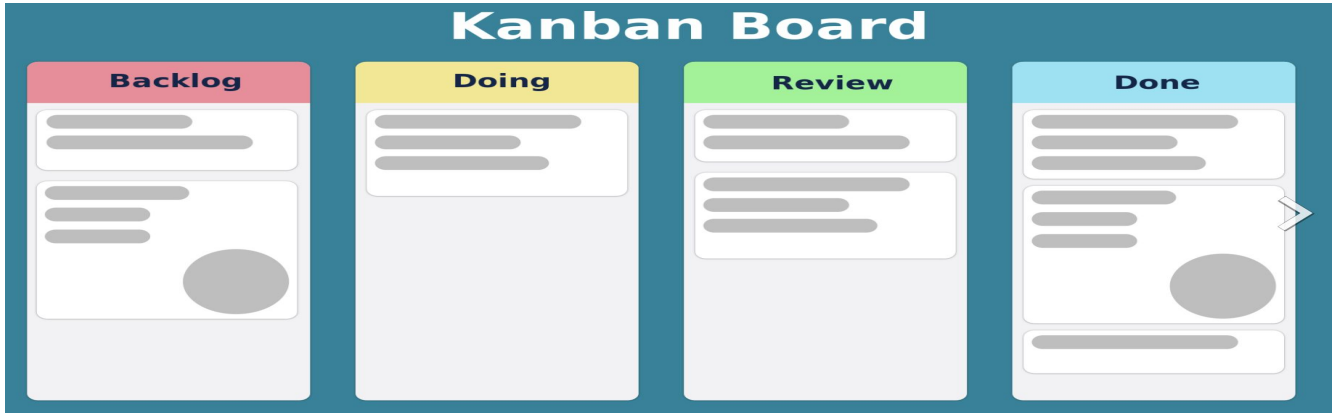
- *Point commun avec l'approche agile: **autonomie**, la "liberté" de déployer en production sans processus strict.*

- *Différence: le principe de se **coordonner** régulièrement avec les porteurs du projet pour bien identifier et prioriser les business, et définir les livrables **("Definition of done").***

Sandy Ingram

Aperçu de quelques approches agiles: Kanban

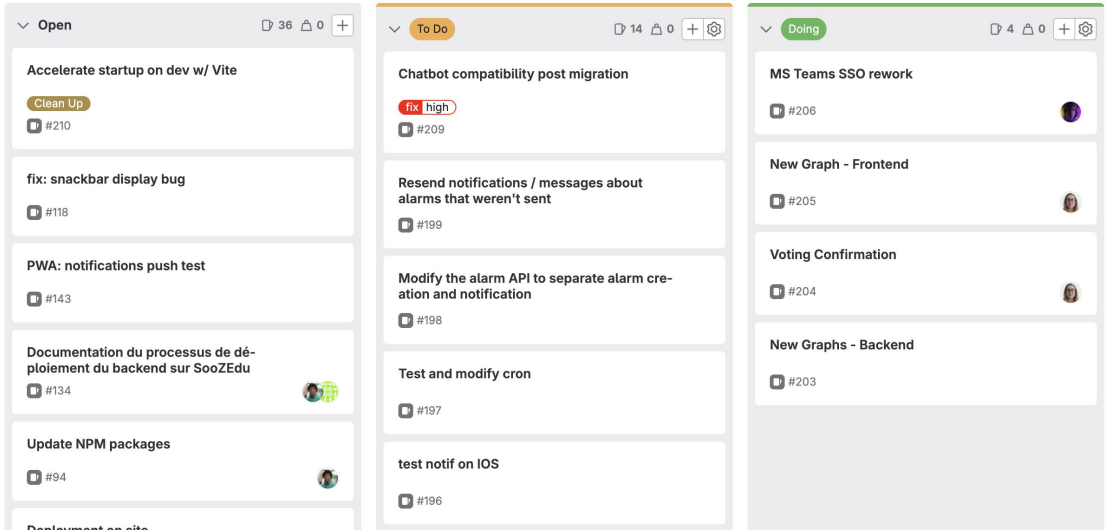
- Visualisation du “workflow” sur un tableau de bord.
- 1 card = 1 user story (mais qu’est ce qu’un user story? …).
- **WIP** work in progress limit = nombre limité de tâche par colonne.
- Minimisation du temps entre les points de “commitment” (engagement) et “completion” (complétion).
- Comme outil, on peut utiliser:
  - un tableau physique avec des “postits”
  - ou un logiciel comme Trello, Gitlab board …



Sandy Ingram

Aperçu de quelques approches agiles: Kanban

- Comme outil, on peut utiliser: un tableau physique avec des “postits” ou un logiciel comme Trello, Gitlab board



(Issue) Board de gitlab pouvant être utilisé comme Kanban board

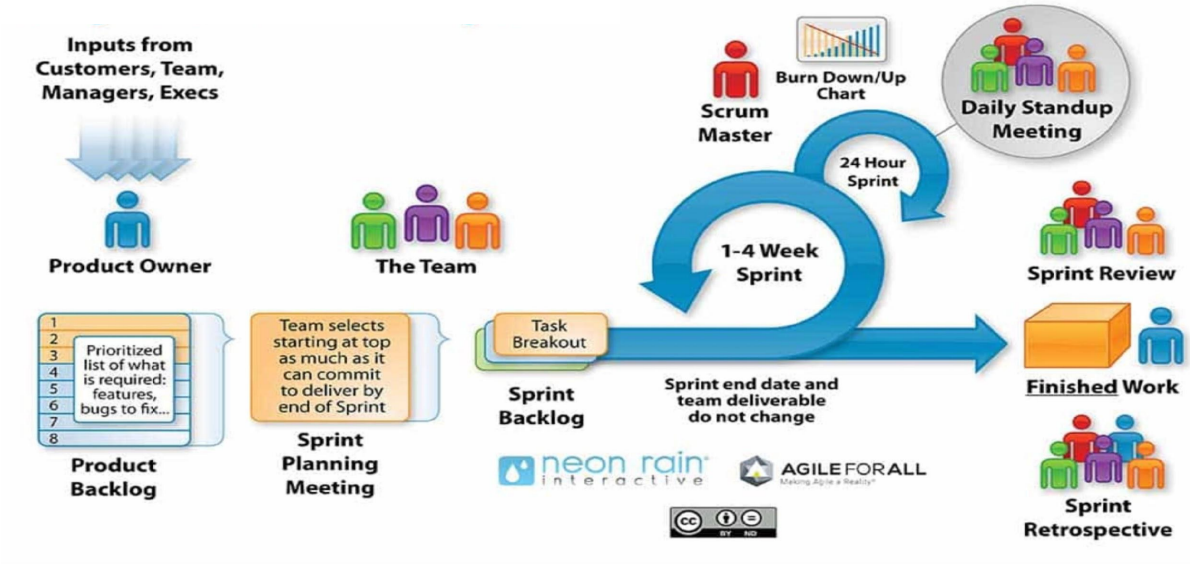
Aperçu de quelques approches agiles: XP

- XP pour “extreme programming”
- Logiciel fonctionnel prime sur une documentation exhaustive, et “releases” régulières.
- Focus sur techniques de programmation en vue d’un logiciel sans “bug”: “pair-programming”, “refactoring”, conventions de codage.
- Est-ce que XP respecte les 12 principes agiles de l’Agile Manifesto?

Aperçu de quelques approches agiles: SCRUM

- Comme XP, SCRUM se concentre sur le développement d'un logiciel fonctionnel et des "releases" fréquents (prototypage rapide) plutôt qu'une documentation exhaustive.
- Alors que XP se concentre sur un logiciel "sans bug" avec des techniques de programmation, SCRUM met le focus sur la "livraison" d'un produit fonctionnel adapté aux besoins avec un processus de développement itératifs.
- Donc SCRUM est plus focalisés sur **le cycle de vie d'un projet et sa gestion.**

Aperçu de quelques approches agiles: SCRUM



https://www.neonrain.com/agile-scrum-web-development/

AUCUN changement n'est permis **pendant** le sprint, mais le sprint peut être annulé s'il devient obsolète.

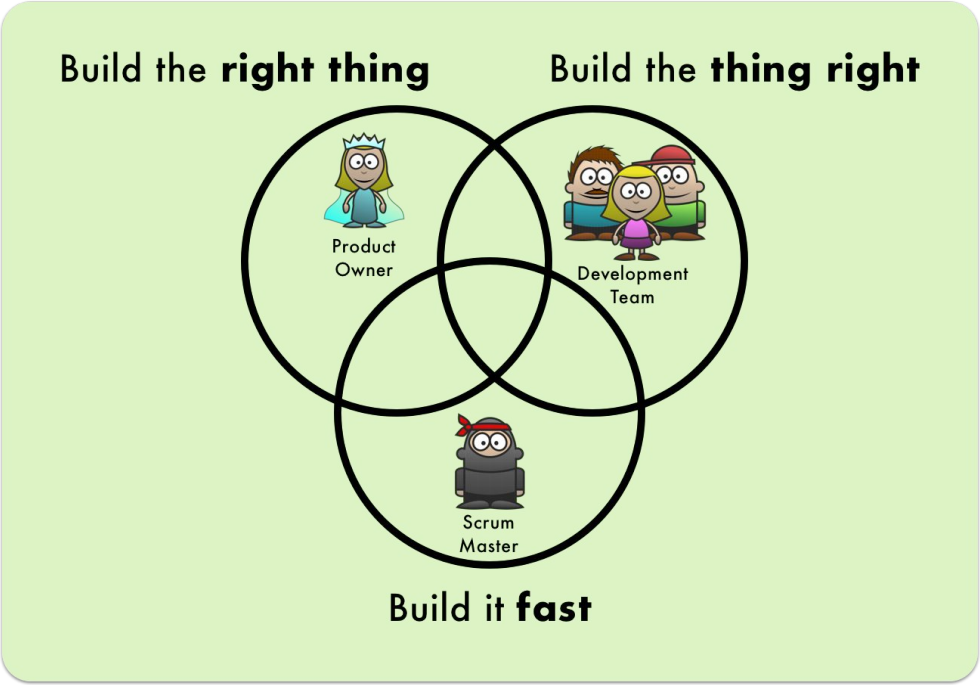
Les Meetings SCRUM

Sandy Ingram

- **Sprint planning:** limité à 8 heures pour un sprint de 4 semaines.
- **Daily Scrum:** <= 15 minutes pour discuter de ce qui a été fait la veille, ce qui est prévu pour le jour-même, et s'il y a des "bottlenecks" ou défis spécifiques.
- **Sprint Review Meeting:** à la fin de chaque sprint.
- **Sprint Retrospective:** à la fin de chaque sprint (limité à 3 heures pour un sprint de 4 semaines).
- **Product Backlog refinement meeting:** séance d'affinage du backlog.

Le SCRUM Team

Sandy Ingram



Le SCRUM Team

Sandy Ingram

- **Product Owner** (PO): identifie les besoins “business”, priorise les items du backlog.
- **Développeurs:** 7+/-2, multidisciplinaires (par ex. pas de silos entre “testeurs” et “développeurs”, **auto-organisés**
- **Scrum Master:** fait le pont entre le PO et les développeurs, and veille sur le respect des “règles” de SCRUM (par exemple la durée d’un daily meeting).
- Les SCRUMS meetings concernent tout le SCRUM Team.

Le SCRUM Team

Sandy Ingram

- **Product Owner** (PO): identifie les besoins “business”, priorise les items du backlog.
- **Développeurs:** 7+/-2, multidisciplinaires (par ex. pas de silos entre “testeurs” et “développeurs”, **auto-organisés**
- **Scrum Master:** fait le pont entre le PO et les développeurs, and veille sur le respect des “règles” de SCRUM (par exemple la durée d’un daily meeting).

La séance de “Sprint Planning”

- Peut se faire avec des cartes “poker” pour estimer les “story” points par user story
- Les estimations se font en terme de durée ou **complexité**.
- lorsque les estimations divergent, on justifie le choix en discutant des tâches pensées et leurs complexités.
- Mais **attention** ni les “user stories” ni les cartes “poker” ne font partie du guide officiel de la méthodologie “SCRUM”.

“Sprint Planning”: estimation des users stories en story points

Sandy Ingram

- Les user stories sont divisées en “tâches” à estimer.
- Les story points représentent soit des: jours idéaux, des heures, ou des “timeless” story points.
- Les Timeless story points estiment la complexité d’une tâche relativement à une autre. C’est moins intuitif qu’une estimation temporelle mais permettent de se focaliser sur la “shipping value” plutôt que le temps de livraison.
- Les tâches individuelles trop complexes doivent être décomposées en sous-tâches pour faciliter l’estimation.

“Sprint Planning”: estimation avec des cartes “poker”

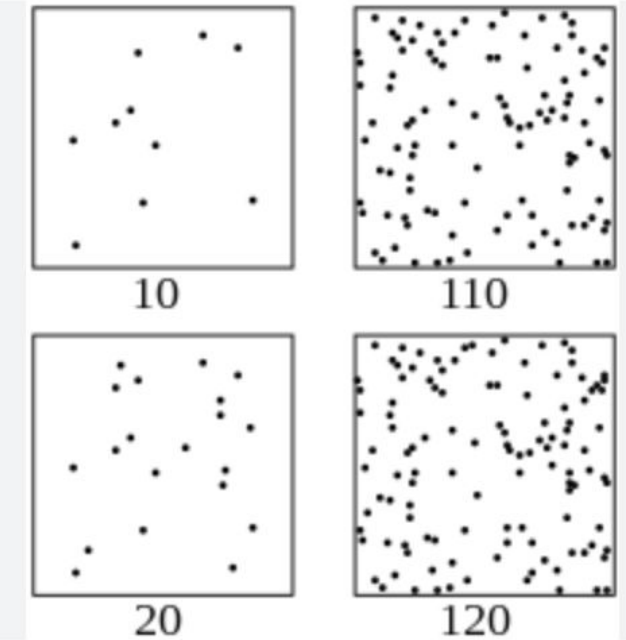
Sandy Ingram

- Chaque développeur fait une estimation personnelle.
- Les estimations sont ouvertes simultanément et comparées.
- Les discussions des estimations divergentes permet une compréhension mutuelle des tâches (qu’ai-je oublié? qu’ai-je sous-estimé ou sur-estimé? y-a-t-il un moyen plus simple à le faire? pourquoi la tâche est si complexe).
- Les estimations sont faites avec des séries, séquences de nombres non-linéaires (Fibonacci, Power of two)
- https://planningpokeronline.com/

Pourquoi utilise-t-on des séries non-linéaires pour les estimations?

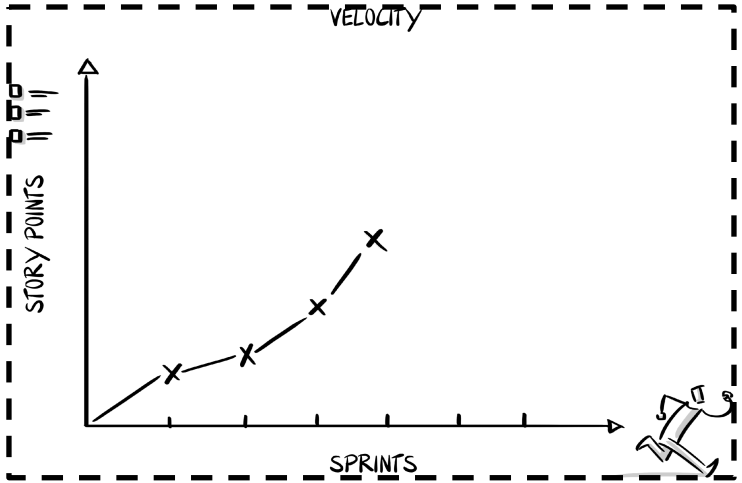
Le changement de 10 to 20 est plus simple à percevoir que celui de 110 to 120.

La perception du changement est influencée par le rapport entre la valeur de départ et l'incrément. Weber's-Fechsner law).



https://en.wikipedia.org/wiki/Weber%E2%80%93Fechner_law

“Retrospective Meeting”: La vélocité



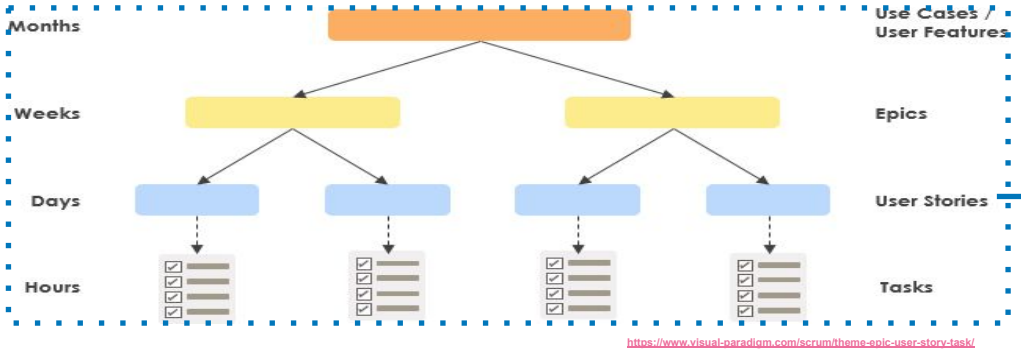
https://www.agile-academy.com/en/scrum-master/velocity-definition-and-how-you-can-calculate-it/

Vélocité moyenne = $$\frac{\sum_{i=1}^{n} (totalStoryPoints * i)}{n}$$

avec n: le nombre de sprints
et totalStoryPoints_i le nombre de of storypoints au sprint I

Les “sprint review meetings” permettent d’identifier la “vélocité” d’une équipe pour améliorer les estimations.

Planification Agile



https://www.visual-paradigm.com/scrum/theme-epic-user-story-task/

Sandy Ingram

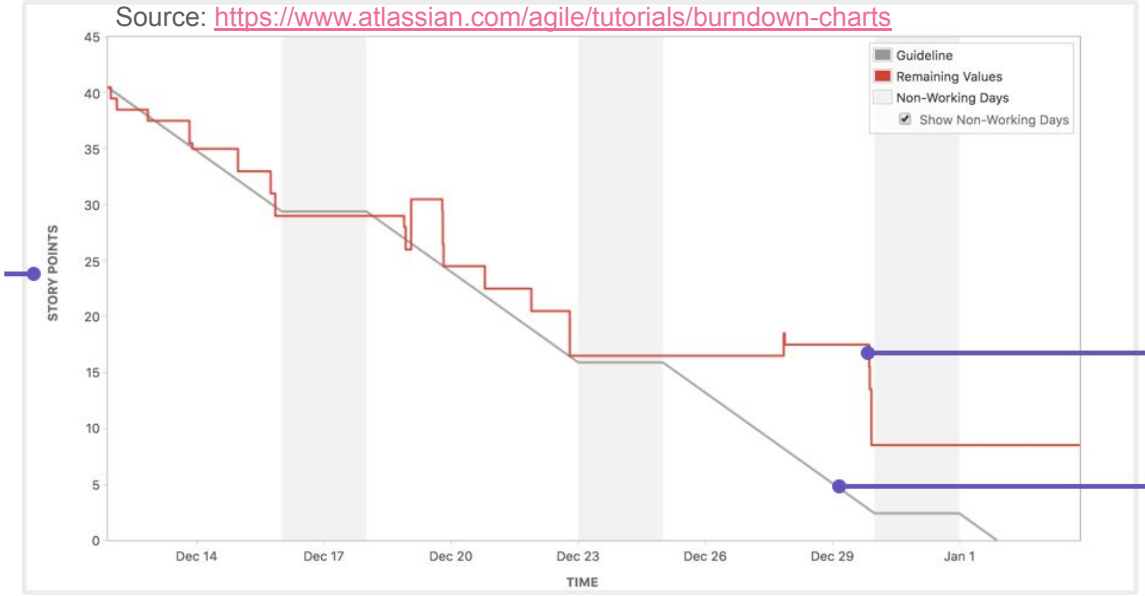
"As a [role], I [want to], [so that]."

Action Benefit

- Le PIB (Product Item Backlog) est découpé en epics, "users stories", et tâches techniques.
- Un epic est accompli en une ou plusieurs itérations.
- Avec SCRUM en particulier:
  - On parle de "sprint" (au lieu d'itérations).
  - Dans le SCRUM guide, on parle évidemment de PIB.
  - mais SCRUM n'impose aucune utilisation de "user stories".

Sprint Burn Down Charts

Source: https://www.atlassian.com/agile/tutorials/burndown-charts



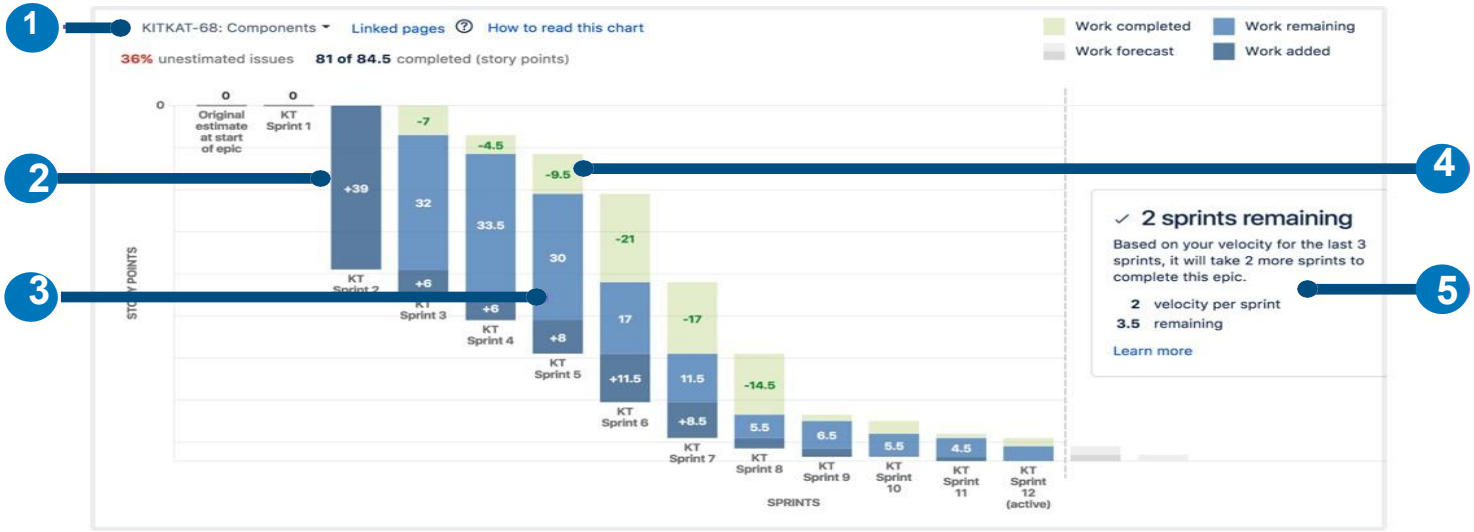
2 **Remaining**
Le travail total restant (story points ou jours idéaux ou heures selon le type d'estimation choisi).

3 **Ligne de référence**
Estimation linéaire basée sur le temps restant

Idéalement, on aimerait que la ligne rouge pense en dessous de la grise

Epic Burn Down Charts

Source: https://www.atlassian.com/agile/tutorials/burndown-charts



1: Epic choisi

2: Nombre de story points ajoutés à l'epic par sprint (après le planning du sprint en question).

3: Le nombre de story points restant pour l'epic choisi.

4: Le nombre de story points accompli par sprint for l'epic choisi.

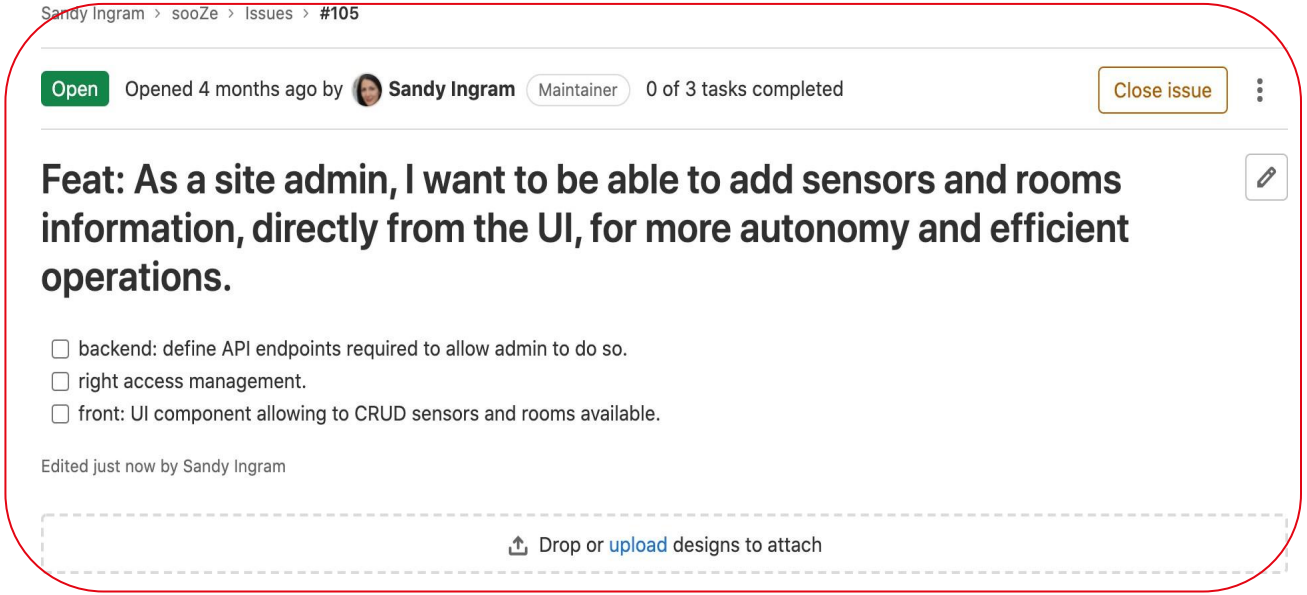
5: (Prédictif) Nombre de sprints restant en fonction de la vélocité de l'équipe.

Utilisation de Gitlab pour une planification agile

Mapping Agile artifacts to GitLab features

Agile artifact	GitLab feature
User story	Issues
Task	Task lists
Epic	Epics
Points and estimation	Weights
Product backlog	Issue lists and prioritized labels
Sprint/iteration	Milestones
Burndown chart	Burndown charts
Agile board	Issue boards

Représentation d'une "user story" avec un gitlab "issue"



"As a [role], I [want to], [so that]."

Action **Benefit**

Applications et systèmes embarqués “SAFE”: Bénéfices de SCRUM

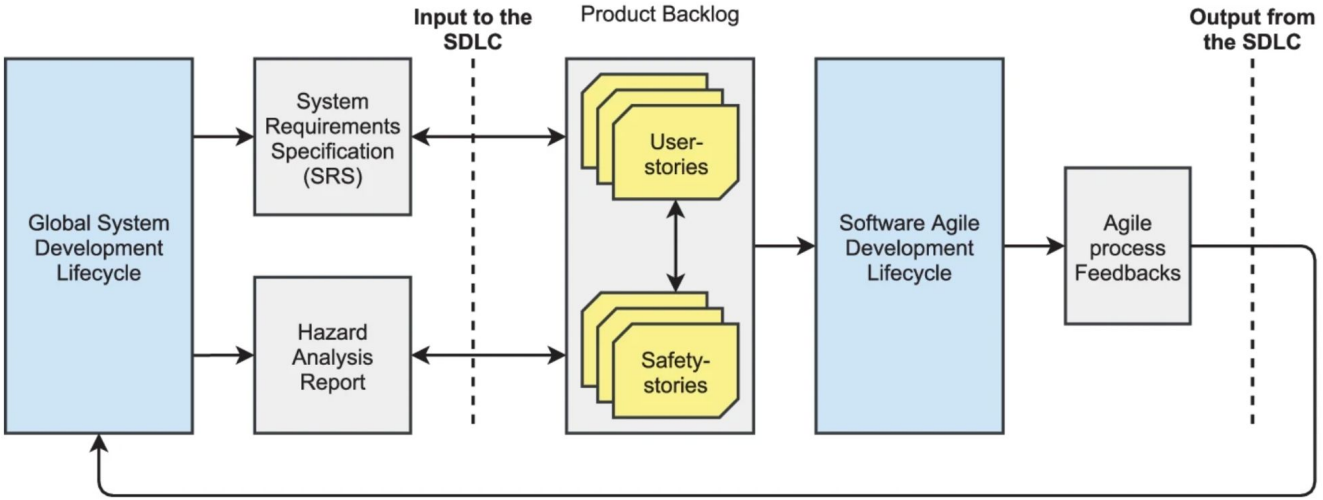
- Développement **itératif**: améliorations incrémentales en gardant le focus sur la sécurité et la fiabilité du système embarqué.
- **Fléxibilité** (se rappeler de Lean) avec le changement de régulations, des besoins du marché.
- Transparence et **coordination** accrue au sein d’équipes **multidisciplinaires** et auto-organisés constitués d’experts du domaine, développeurs de logiciels, de hardware, assurance qualités (daily SCRUM, sprint planning)
- **Priorisation** des “items” dans le PIB en fonction de la valeur “business” (e.g. ceux critiques pour la sécurité)
- Amélioration **continue** (sprint retrospective).

Applications et systèmes embarqués “SAFE”: Limites et adaptations de SCRUM

- Processus de test **rigoureux**
  - ceci est cohérent avec le principe agile de logiciel fonctionnel mais nécessiterait des cycles de développements plus **longs**.
  - Intégration de pratiques DevOps.
- Rien de spécifique dans le guide SCRUM par rapport à une méthodologie de gestion des risques.
- Au delà du côté fonctionnel, il y a un besoin de **documentation plus “exhaustive”** (conformité à des standards ISO, certifications, et réglementations spécifiques au domaine: aéorospace, médical, automobile).

S4S: Scrum for Safety

From: Scrum for safety: an agile methodology for safety-critical software systems

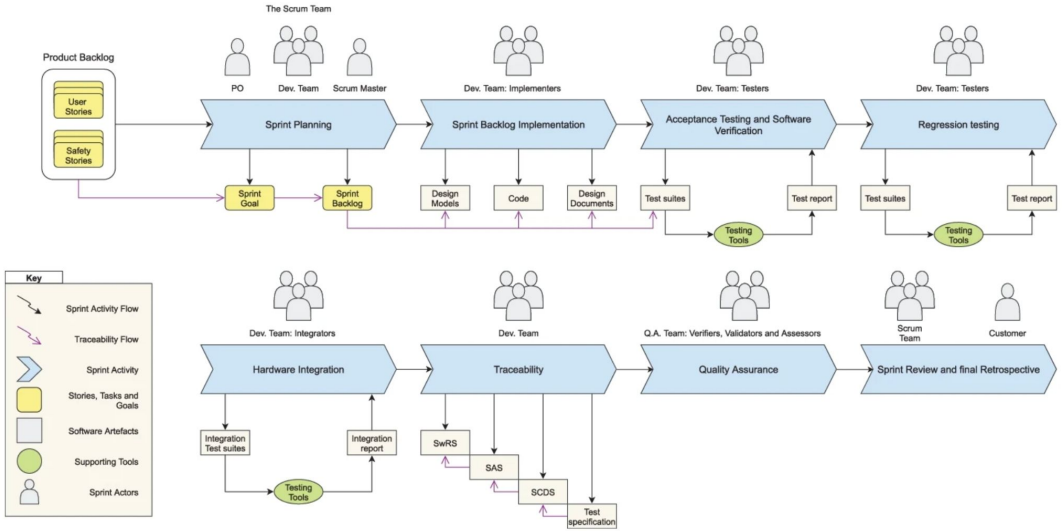


S4S integration with the global system life-cycle

https://link.springer.com/article/10.1007/s11219-022-09593-2

S4S: Scrum for Safety

From: Scrum for safety: an agile methodology for safety-critical software systems



S4S Safe-Sprint Structure

https://link.springer.com/article/10.1007/s11219-022-09593-2

Synthèse

- Deux piliers du génie logiciel: DevOps et méthodologie agile.
- Agile Manifesto:
  - 4 valeurs principales et 12 principes
  - des méthodologies agiles complémentaires: Kanban, XP, SCRUM
  - SCRUM en détail: Scrum team, 5 meetings, planification
  - Applicable aux applications “SAFE” et systèmes embarqués en complément avec d’autres approches.

Passons à la pratique

- Constituer des groupes
  - les groupes du cours systèmes embarqués s'annoncent et cherchent d'autres membres (feuille excel à remplir en live).
- Créer un "groupe" privé gitlab : 25_softweng_xxx_yyy_zzz avec la série xxx_yyy_… représentant les 3 premières du nom de famille de chaque membre du groupe
- Nous y inviter en tant que "maintainer"
- Repérer le git "issue board" sous plan.

Références

- Agile Manifesto: https://agilemanifesto.org/history.html
- Official Scrum Guide: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100 (Web version: https://scrumguides.org/scrum-guide.html),
- Planning Agile Development with gitlab: https://about.gitlab.com/blog/2018/03/05/gitlab-for-agile-software-development/
- Sprint Planning: https://www.scrum.org/resources/what-is-sprint-planning
- Scrum Metrics (Velocity, Capacity, Burndown charts): https://www.atlassian.com/agile/scrum/scrum-metrics (not inherent or imposed by Scrum).
- Scrum and Epic **Burn down charts**: https://www.atlassian.com/agile/tutorials/burndown-charts
- **Estimation with storypoints**: https://www.atlassian.com/agile/project-management/estimation
- Scrum Glossary: https://www.scrum.org/Resources/Scrum-Glossary
- **StoryPoints revisited** https://ronjeffries.com/articles/019-01ff/story-points/Index.html
- **Cumulative Flow Diagram**: https://knowledgebase.kanbanize.com/hc/en-us/articles/360015034420-The-Cumulative-Flow-Diagram-CFD-
- **Scrum for Safety:** https://link.springer.com/article/10.1007/s11219-022-09593-2

Lecture Optionnelle

- https://www.puppet.com/system/files/2013-State-of-Devops-Report.pdf
- 2013 State of Devops by Puppet Labs https://www.puppet.com/system/files/2013-State-of-Devops-Report.pdf
- History of Agile development: https://techbeacon.com/app-dev-testing/agility-beyond-history-legacy-agile-development
- Iterative Software Development: https://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf
- Historical Roots of Agile Methods: Where did “Agile Thinking” Come from? https://link.springer.com/chapter/10.1007/978-3-540-68255-4_10
- **ScrumBan** vs Scrum vs Kanban (nice comparative table!) : https://ora.pm/blog/scrum-vs-kanban-vs-scrumban/ (PS: ”The product owner push and assigns tasks to team members” => not 100% correct).
- ScrumBan https://www.productplan.com/glossary/scrumban/