Génie logiciel : Méthodes agiles

Arnaud Labourel (arnaud.labourel@univ-amu.fr)

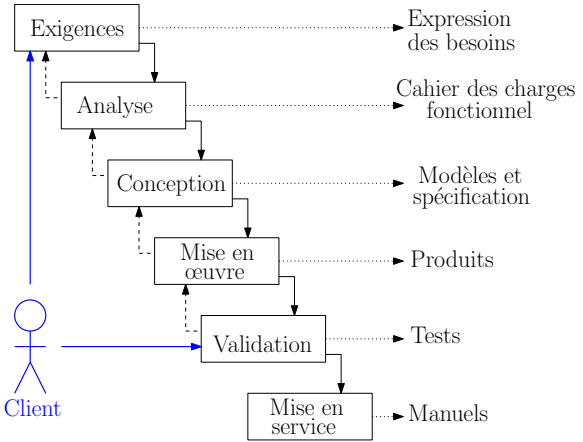
25 octobre 2024

amU Faculté des sciences Aix Marseille Université

Section 1

Les raisons de l'agile

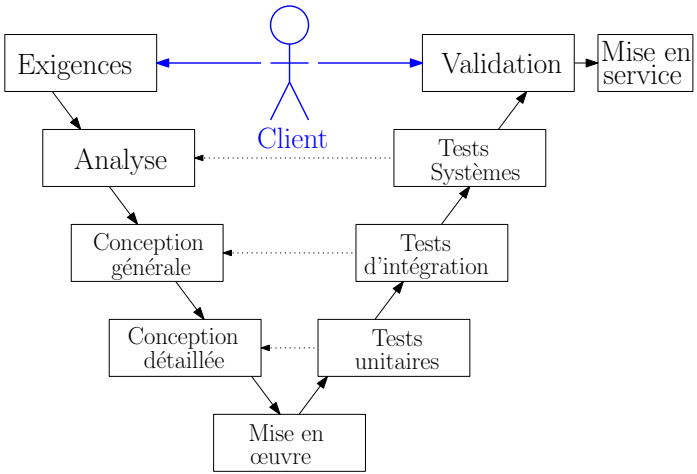
Les défauts du modèle en cascade



Défauts de Cascade

Exigences définies au début du projet (potentiellement incomplètes)
Client peu impliqué dans le projet (seulement au début et à la fin)
Cycle en trois temps peu adapté au logiciel
Analyse $\rightarrow$ Conception $\rightarrow$ Implémentation
(souvent les difficultés n'apparaissent qu'à l'implémentation)

Les défauts du modèle en V



Défauts du cycle en V

Globalement les mêmes que Cascade :

- client peu sollicité ;
- séparation en phase peu adaptée au développement de logiciel

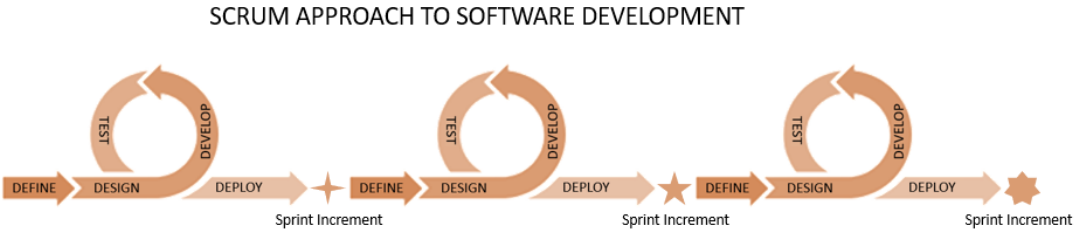
Risques des approches séquentielles et linéaires de phases

- Dépendances par rapport aux exigences identifiées en début de projet. Les exigences peuvent être :
  - **pas assez précises** : le client n'arrive pas vraiment à préciser ces besoins de manière précise ;
  - **instables** : le client peut changer d'avis ;
  - **infaisables ou trop complexe** : il peut arriver que certaines fonctionnalités demandées par le client soient trop difficile à mettre en œuvre.
- Beaucoup de temps dédié à la planification :
  - Quasi-impossibilité d'estimer correctement la durée de tâches complexes
  - N'apporte pas vraiment de valeur ajoutée pour le client

$\Rightarrow$ Utiliser des méthodes incrémentales et itératives

- Enchaîner des mini-cycles en V sur différentes fonctionnalités du logiciel
- Utiliser une méthodologie agile : Scrum, XP, Kanban, ...

Approche cascade vs agile scrum



Approche “agile” 2001

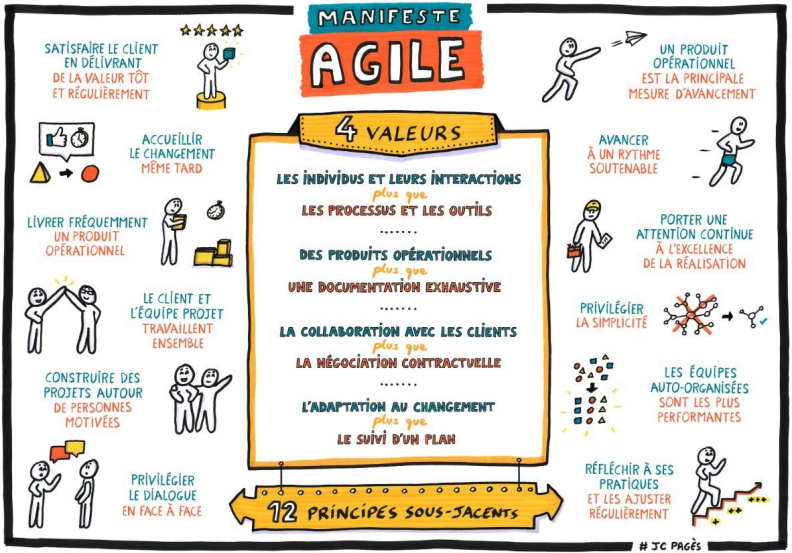
2001 : manifeste agile signé par 17 développeurs

- Impliquer le client de manière continu et faire attention à ses besoins ;
- Redonner aux développeurs l'organisation du projet ;
- Mettre de la souplesse dans la gestion du projet (cycle de production itératif et incrémental) ;
- Remettre l'humain au centre du processus de développement logiciel ;
- Limiter au maximum les tâches de planification pour se concentrer sur activités produisant de la valeur ajoutée.
- Mettre en place des processus d'amélioration continue

Section 2

Manifeste agile

Manifeste agile



Quatre valeurs

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

Les seconds éléments ont de la valeur, mais on privilégie les premiers éléments.

Douze principes (1/3)

- Notre plus haute priorité est de **satisfaire le client** en **livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée**.
- **Accueillez positivement les changements de besoins**, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.
- **Livrez fréquemment un logiciel opérationnel** avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
- Les **utilisateurs** ou leurs représentants et les **développeurs** doivent **travailler ensemble quotidiennement** tout au long du projet.

Douze principes (2/3)

- Réalisez les projets avec des personnes motivées. **Fournissez-leur l'environnement et le soutien dont ils ont besoin** et faites-leur confiance pour atteindre les objectifs fixés.
- La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le **dialogue en face à face**.
- Un **logiciel opérationnel** est la principale **mesure d'avancement**.
- Les processus Agiles encouragent un **rythme de développement soutenable**. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.

Douze principes (3/3)

- Une attention continue à l'**excellence technique** et à une **bonne conception** renforce l'Agilité.
- La **simplicité**, c'est-à-dire l'art de **minimiser** la quantité de **travail inutile**, est essentielle.
- Les meilleures architectures, spécifications et conceptions émergent d'**équipes autoorganisées**.
- À intervalles réguliers, l'équipe **réfléchit aux moyens de devenir plus efficace**, puis règle et modifie son comportement en conséquence.

Nombreuses variantes et sous-variantes de framework agile

- **Scrum** : accent sur la gestion d'équipe et de projet
- **Kanban** : accent sur la visualisation du flux de travail
- **XP programming** : accent sur les pratiques de développement
- **Programmation pilotée par le comportement (BDD)** : accent mis sur les besoins client et la spécification via des exemples
- ...

Nombreuses variantes et différentes pratiques selon :

- les besoins et contraintes de l'entreprise
- les habitudes et les compétences des membres de l'équipe

Possibilité d'utiliser des concepts de différentes méthodologies agiles.

Section 3

Scrum

Définition de Scrum

Cadre de travail léger qui a besoin d'un *Scrum Master* pour favoriser un environnement où :

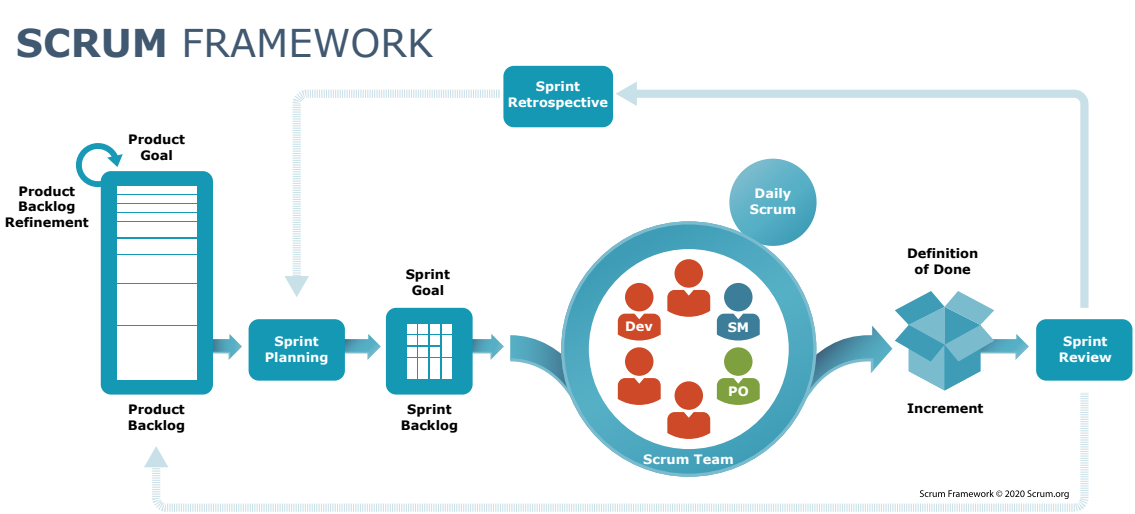
1. Un *Product Owner* ordonne le travail à faire pour résoudre un problème dans le *Product Backlog*.
2. La *Scrum Team* transforme une sélection de ce travail en un ou plusieurs *Increment* (ensemble de fonctionnalités) lors d'un *Sprint*.
3. La *Scrum Team* et ses parties prenantes inspectent les résultats et s'adaptent pour le prochain *Sprint*.
4. On Répète les étapes ci-dessus jusqu'à la complétion du projet.

Base de Scrum

Scrum est basée sur :

- l'**empirisme** : la connaissance provient de l'expérience et la prise de décision s'appuie sur l'observation de faits
- une pensée **lean** (dégraissé) : réduire le gaspillage et se focaliser sur l'essentiel
- une approche **itérative et incrémentale** pour optimiser la prédictibilité et le contrôle de risque
- un idéal d'**amélioration permanente** : toujours réfléchir à comment s'améliorer et améliorer la productivité de l'équipe

Fonctionnement de scrum



Sprint et product backlog

Sprint : unité de base de temps pour l'avancement du projet (1 à 4 semaines) :

- Période de travail encadrée pour livrer une partie du produit ;
- Axé sur un ou des objectifs spécifiques définis en début de Sprint.

Pour pouvoir définir les objectifs, il faut maintenir à jour une liste des besoins : le **product backlog**

Product backlog

Liste ordonnée de toutes les fonctionnalités à implémenter.

Évènement lors d'un sprint

- **Sprint Planning** (planification de Sprint) au début du sprint (quelques heures max) :
  - Au début de chaque sprint, l'équipe sélectionne le travail à effectuer à partir du *Product Backlog*.
  - Définit l'objectif du sprint et le plan pour y parvenir.
- **Daily Scrum** (mêlée quotidienne) une fois par jour ouvré du sprint (15 minutes) :
  - Communiquer les tâches réalisées du jour précédent
  - Se répartir les tâches à traiter pour la journée
  - Identifier les difficultés/obstacles
- **Sprint Review** (revue de Sprint) à la fin du sprint :
  - L'équipe présente le travail accompli pendant le sprint.
  - Les parties prenantes (équipes de développement, clients, ...) donnent leur retour.
- **Sprint Retrospective** (rétrospective de Sprint) après la fin du sprint :
  - L'équipe réfléchit au processus du sprint et identifie des améliorations.
  - Focalisé sur l'amélioration continue.

Forme du product backlog

Pas véritablement de format décrit par les concepteurs de scrum mais souvent on utilise la notion de récits utilisateur (*User Story*) issu de XP.

Un récit utilisateur : phrase simple permettant de définir le contenu d'une fonctionnalité à développer.

Format

En tant que **qui**, je veux **quoi** afin de **pourquoi** (plus **critères d'acceptation**)

- le **qui** : utilisateur associé à un rôle exprimant le besoin (élève, enseignant, bibliothécaire, ...)
- le **quoi** : description succincte du comportement attendu
- le **pourquoi** : permet d'identifier l'intérêt de la fonctionnalité et sa valeur ajoutée
- les **critères d'acceptation** : liste d'éléments permettant à l'utilisateur de confirmer que la fonctionnalité répond à ces attentes

Exemple de récit utilisateur (1/2)

En tant que vendeur en succursale, **je veux** pouvoir rechercher mes clients par leur prénom et leur nom de famille **afin de** les retrouver rapidement lorsque je reçois un appel de leur part.

Critères d'acceptation :

- je peux rechercher par prénom sans le nom de famille
- je peux rechercher par nom de famille sans le prénom
- je peux rechercher par prénom et nom de famille en même temps
- je peux avoir des suggestions proches si je fais une erreur en écrivant

Exemple de récit utilisateur (2/2)

En tant qu'élève/apprenant, **je veux** voir mon score et la correction de mes mauvaises réponses dès que j'ai validé mon exercice **afin de** voir mes erreurs et pouvoir m'améliorer tout de suite.

Critères d'acceptation :

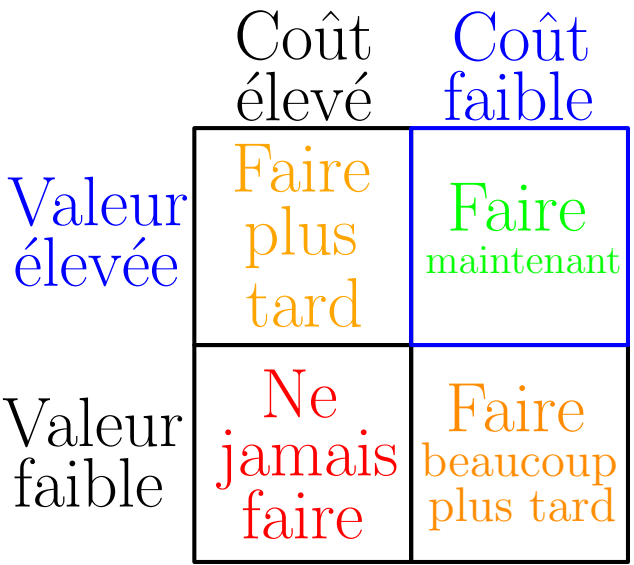
- Lorsque je valide mon exercice, je vois mon score et la liste des questions avec le score de chacune ;
- Je peux naviguer et revoir les questions avec mes réponses ;
- Mes réponses erronées sont indiqués comme fausse et la bonne réponse est affichée ;
- Je peux aussi choisir de naviguer uniquement à travers les réponses erronées.

Caractéristiques d'un récit utilisateur

L'acronyme INVEST est souvent utilisé pour lister des principales qualités d'un bon récit utilisateur :

- I pour **Indépendant** : la description du récit est indépendante des autres pas de dépendance avec les autres récits ;
- N pour **Négociable** : le contenu détaillé n'est pas gravé dans le marbre, mais peut être négocié avec l'utilisateur ;
- V pour **Valeur** : la fonctionnalité a de la valeur pour l'utilisateur ;
- E pour **Estimable** : le travail pour réaliser le récit peut être estimé ;
- S pour **Small** (petit) : le récit doit pouvoir être réalisé au cours d'un Sprint ;
- T pour **Testable** : les critères d'acceptation doivent être vérifiables en pratique.

Sprint planning : définir le but du sprint (1/2)



- Choisir les récits sur lesquels travailler
  $\Rightarrow$ analyse coût/valeur ajoutée
- Expliquer à l'équipe l'utilité du travail prévu dans le sprint

$\Rightarrow$ permet de définir un ou plusieurs *Increments* (ensemble de fonctionnalités ajoutant de la valeur et pouvant être validé).

Product Owner

Membre de l'équipe qui doit :

- maximiser la valeur du produit
- gérer le *Product Backlog*

Sprint planning : décider comment faire (2/2)

Une fois les récits choisis pour le sprint, il faut :

- les décomposer en tâches faisables en une journée
- définir les conditions pour lesquelles l'*increment* est validé : *Definition of Done* (DoD) :

$\Rightarrow$ Rédaction du Sprint Backlog : objectif du Sprint, liste des récits utilisateurs à traiter, plan d'action (découpage en tâches)

Definition of Done

Standard de qualité nécessaire pour valider un *increment* (Tests, Revue de code, retour utilisateur, ...)

Fin du sprint

Sprint Review

Faire le bilan du travail réalisé durant le sprint :

- mettre à jour l'état d'avancement des fonctionnalités du *product backlog*
- livraison des incréments pour un retour
- adaptation du *product backlog* en fonction des retours

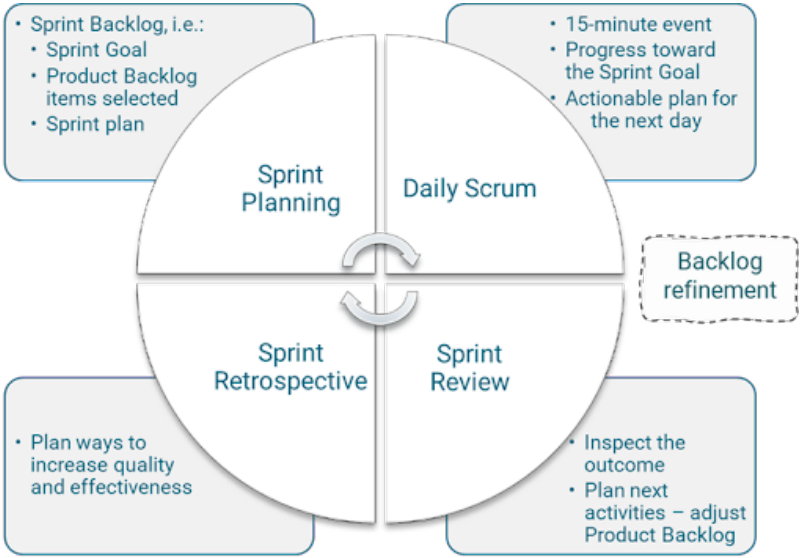
Sprint Retrospective

Inspecter le déroulement du Sprint et créer un plan d'amélioration à adopter pour les prochains Sprints.

Processus d'amélioration continu :

- Questionnement sur l'efficacité des outils et des processus ;
- Discussions sur les difficultés rencontrées lors du sprint et comment les éviter.

Résumé événements scrum



Scrum team

Une équipe dans le cadre de scrum est composée :

- d'un *product owner*
- d'un *scrum master*
- d'un nombre quelconque de *developers*

Taille idéale d'équipe : 5-9 membres au total

Avoir une équipe trop grande pose trop de difficultés :

- le nombre de canaux de communication augmentent de manière quadratique avec le nombre de membres ;
- pour une équipe trop grande, il devient impossible pour les membres de retenir qui fait quoi.

$\Rightarrow$ il vaut mieux découper une équipe avec trop de membres en plusieurs équipes.

Developers

Les **Developers** doivent :

- Créer un plan de *Sprint*, un *Sprint Backlog* ;
- Inculquer la notion de qualité en adhérant à une *Definition of Done* ;
- Adapter leur plan chaque jour par rapport à l'Objectif de *Sprint* ;
- Se tenir mutuellement responsables en tant que professionnels.

Ils peuvent avoir des compétences variées et complémentaires, mais ils ne doivent pas se cantonner à un seul rôle.

Product Owner

Le **Product Owner** :

- est redevable de maximiser la valeur du produit résultant du travail de la Scrum Team (prioriser les récits utilisateurs en fonction de leurs valeurs et coût)
- doit gérer le *Product Backlog* (seule personne pouvant le modifier) ce qui inclut :
  - Formuler et communiquer explicitement l'Objectif de Produit ; Créer et communiquer clairement les éléments du *Product Backlog* ;
  - Ordonner les éléments dans le *Product Backlog* ; et
  - S'assurer que le *Product Backlog* est transparent, visible et compris.

$\Rightarrow$ Il est important que ses décisions soient respectées par les membres de l'équipe.

Scrum Master

Le **scrum master** aide l'équipe :

- pour la mise en place de Scrum tel que défini dans le Guide Scrum
- à se focaliser à créer des Increments de grandes valeurs
- à ce que les différents événements scrum (planning, daily, review, retrospective) se fassent et ne durent pas trop longtemps

Le *Scrum Master* aide le *Product Owner* sur la gestion du Product Backlog ;

Rôle de facilitateur et de management.

Bilan des Artefacts de Scrum

Les artefacts de Scrum représentent un travail ou une valeur. Ils sont conçus pour maximiser la transparence des informations clés.

- Backlog Produit :
  - Liste ordonnée de toutes les tâches à faire (fonctionnalités, correctifs, tâches techniques).
  - Affiné après chaque Sprint
  - Géré par le *Product Owner*.
- Backlog de Sprint :
  - Liste des tâches que l'équipe s'engage à réaliser dans le sprint actuel.
  - Géré par l'Équipe de Développement.
- Incrément :
  - La somme de tous les éléments du Backlog Produit complétés à la fin d'un sprint.
  - Représente un produit potentiellement livrable.

Forces et faiblesses de Scrum

Avantages de Scrum

- **Transparence** : Visibilité claire des progrès.
- **Adaptabilité** : Capacité à réagir rapidement aux changements.
- **Satisfaction client** : Retours réguliers des parties prenantes.
- **Concentration**: Les équipes se concentrent sur les tâches prioritaires.
- **Amélioration continue** : Les rétrospectives régulières stimulent les améliorations.

Défis de Scrum

- **Discipline de l'équipe** : Nécessite un engagement envers les pratiques.
- **Engagement des parties prenantes** : Les parties prenantes doivent être régulièrement impliquées.
- **Évolutivité** : Difficile de mettre en œuvre Scrum à grande échelle.
- **Clarté des rôle**s : Une compréhension claire des rôles est essentielle au succès.

Section 4

Déroulement dernier projet

Organisation du projet

- Développement d'un logiciel ;
- Équipes de 4 étudiants ;
- Méthodologie scrum ;
- Planning des sprints :
  - sprint 0 : 25 oct-7 nov
  - sprint 1 : 8 nov-21 nov
  - sprint 2 : 22 nov-5 déc
  - sprint 3 : 6 déc-12 déc (13 déc. soutenances)
- rendus des artefacts scrum à chaque fin de sprint

Section 5

Kanban

Introduction à Kanban

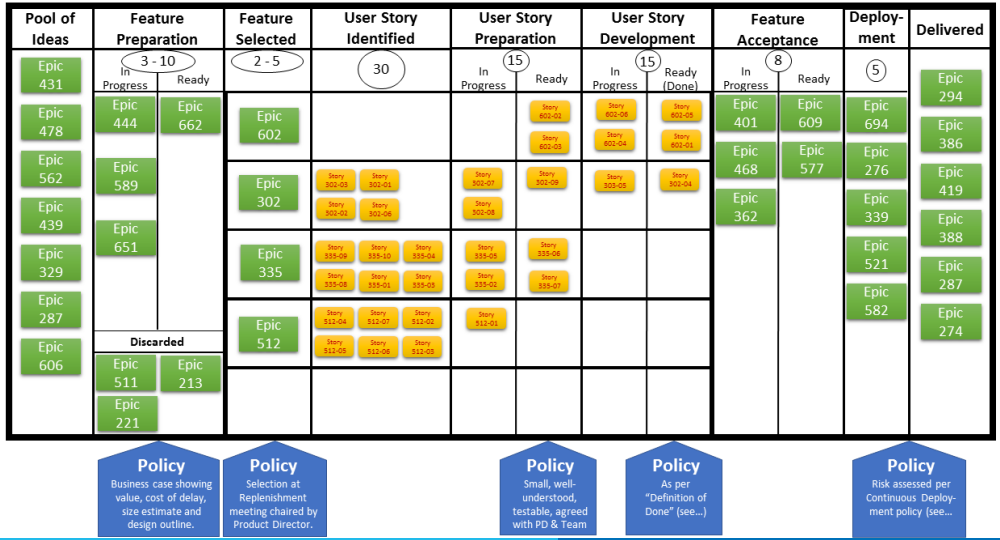
- Kanban :
  - Méthode Agile de gestion visuelle du flux de travail.
  - Origine : Système de production de Toyota (1950).
  - Objectif : Améliorer l'efficacité en visualisant le processus, limitant le travail en cours (WIP), et optimisant le flux.
  - Terme *Kanban* ayant le sens de carte de signalisation
- Caractéristiques principales :
  - Visualisation des tâches via un tableau de tâches.
  - Limitation du nombre de tâches en cours.
  - Flux de travail continu.

Tableau Kanban simple

Le Tableau Kanban

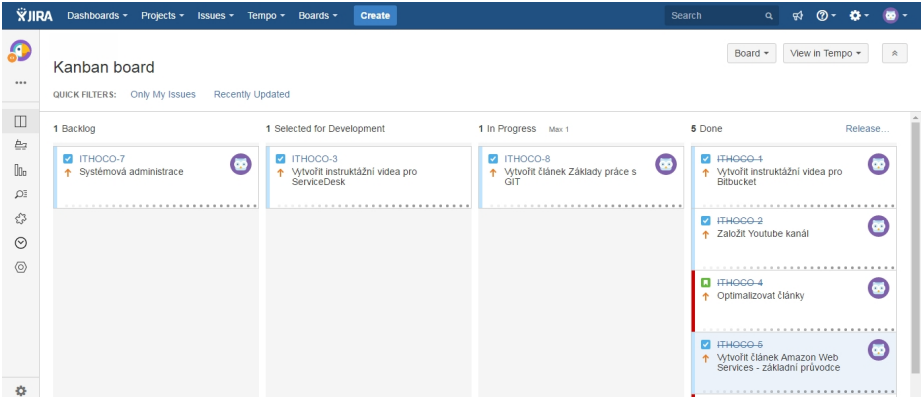
- Colonnes (état d'avancement des tâches), généralement il y a au moins trois colonnes :
  - À faire : Liste des tâches (*user stories* ou *epics*) à prendre en charge.
  - En cours : Tâches en cours de traitement avec une limite Work In Progress (WIP).
  - Terminé : Tâches complétées et prêtes à être livrées.
- Cartes :
  - Chaque tâche est représentée par une carte.
  - La carte contient des informations essentielles comme la description de la tâche, la priorité, le responsable, ...
- *User Stories* : récits utilisateurs
- *Epics* (Épopées) :
  - ensemble de tâches cohérentes visant à un même but
  - peu nombreuses : quelques *epics* par mois vs dizaines d'*user stories* par mois

Tableau Kanban complexe



Kanban logiciel

Il existe des logiciels spécifiques de suivi du travail : Agile SAP, Kanban Tool, Jira Agile, Kanboard, Taiga, . . .



Principes Kanban

- **Commencer par ce que vous faites actuellement** : les rôles et processus déjà définis restent même s'ils peuvent évoluer
- **Accepter d'appliquer les changements évolutifs et augmentés** :
  - les changements continus, augmentés et évolutifs sont le moyen d'améliorer le système
  - changements de petite envergure limitant les risques d'échecs
- **Respecter le processus actuel, les rôles, les responsabilités et les titres**
- **Favoriser le leadership à tous les niveaux**

Les Pratiques Clés de Kanban

- Visualiser le flux de travail :
  - Utilisation d'un tableau Kanban pour suivre les tâches à travers différentes étapes.
  - Permet à toute l'équipe de comprendre l'état du travail.
- Limiter le travail en cours (Work in Progress WIP) :
  - Définir des limites pour chaque colonne (par ex., nombre maximum de tâches).
  - Empêche la surcharge et améliore l'efficacité.
- Gérer le flux :
  - Surveiller et optimiser le mouvement des tâches dans le système.
  - Objectif : un flux de travail fluide et sans blocage.
- Rendre les processus explicites :
  - Clarifier et définir les règles et les processus.
  - Aider l'équipe à comprendre et suivre des étapes spécifiques pour chaque tâche.
- Boucles de rétroaction :
  - Organiser des réunions régulières pour améliorer le processus.
  - Discussions sur les goulots d'étranglement et les améliorations possibles.

Avantages et Défis de Kanban

Avantages

- Flexibilité : ajouts ou retraits des tâches à tout moment.
- Amélioration continue : révision et une optimisation continues du processus.
- Visualisation claire : l'état de chaque tâche est visible, ce qui permet une gestion efficace.
- Réduction des goulots d'étranglement : Le suivi du flux et des WIP limite les blocages.
- Meilleure collaboration : Encourage une communication continue.

Défis

- Risque de surcharge si les WIP ne sont pas respectés.
- Peut manquer de structure formelle pour les équipes moins disciplinées.
- Nécessite une culture d'amélioration continue.

Section 6

Bilan méthodologie agile

Bilan méthodologie agile

Avantages de l'agile

- Adaptation rapide au changement
- Autogestion de petite équipe de développement qui peut être valorisante
- efficacité augmentée (moins de perte de temps de planification)

Désavantages de l'agile

- méthodologie dévoyée et mal appliquée
- certaines pratiques demandent un certain niveau
- peu adapté à des projets d'envergure :
  - grosses équipes (100+ personnes)
  - longues durées (plusieurs années)

  $\Rightarrow$ nécessité de modèles pour faire travailler ensemble plusieurs équipes agiles (SAFe, LeSS, Scrum@Scale, ...)