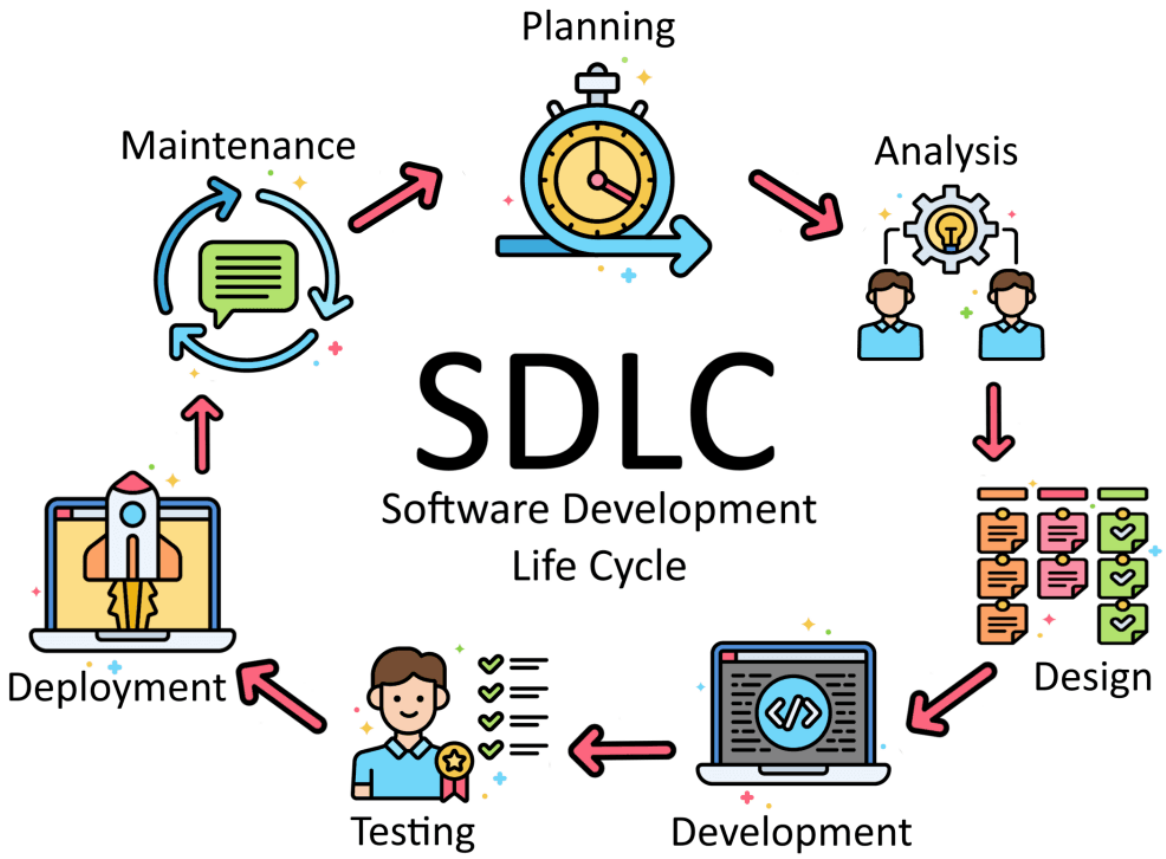
Gestion de Projet Agile



Séance 1 : Introduction et Contexte

- Pourquoi l'agilité ? Limites des méthodes traditionnelles (cycle en V)
- Histoire et manifeste agile (2001)
- Les 4 valeurs et 12 principes du manifeste agile

- **TD/Pratique**:
  - Quiz interactif sur les principes agiles
  - Discussion : "Quels problèmes avez-vous rencontrés en gestion de projet ?"

Séance 2 : Scrum - Fondamentaux

- Introduction à Scrum : définition et philosophie
- Les 3 rôles : Product Owner, Scrum Master, Équipe de développement
- Les 3 artefacts : Product Backlog, Sprint Backlog, Incrément

- **TD/Pratique:**
  - Étude de cas : identifier les rôles dans un projet fictif

Séance 3 : **Scrum - Cérémonies et Flux**

- Les 5 événements Scrum : Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective, Le Sprint lui-même
- Durées recommandées et objectifs de chaque cérémonie

- **TD/Pratique:**
  - Simulation d'un Daily Scrum (mise en situation)

Séance 4 : **User Stories et Product Backlog**

- Concept des User Stories : format "En tant que... je veux... afin de..."
- Critères d'acceptation (Given/When/Then)
- Priorisation du backlog (MoSCoW, valeur métier vs effort)

- **TD/Pratique:**
  - Atelier : Rédiger 5-6 user stories pour une application étudiante (ex: appli de gestion de cours)

Séance 5 : **Estimation et Planification Agile**

- Estimation relative (story points) vs estimation absolue
- Techniques : Planning Poker, T-shirt sizes, Bucket system
- Velocity et prédictibilité

- **TD/Pratique:**
  - Atelier Planning Poker avec des scénarios concrets

Séance 6 : **Tableau Kanban et Flow**

- Origine du Kanban (Toyota) et adaptation logicielle
- Les 6 pratiques du Kanban method
- WIP limits et optimisation du flux
- Différences Scrum vs Kanban

- **TD/Pratique:**
  - Création d'un tableau Kanban physique pour un projet fictif

Séance 7 : **Outils de Gestion Agile**

- Présentation des outils populaires : Jira, Trello, Azure DevOps, Linear
- Critères de choix selon le contexte projet

- **TD/Pratique:**

- TP guidé : Création d'un projet dans un outil (Jira, ClickUp ou Trello)
- Configuration d'un backlog, création de tickets, simulation d'un sprint

Séance 8 : Lean, XP et Autres Pratiques Agiles

- Extreme Programming (XP) : TDD, pair programming, intégration continue
- Lean Software Development : 7 types de gaspillage
- DevOps et Continuous Delivery
- SAFe, LeSS (introduction légère aux frameworks à grande échelle)

- **TD/Pratique:**
  - Analyse comparative : quelle méthode pour quel contexte ?

Séance 9 : Métriques Agiles et Reporting

- Burndown/Burnup charts
- Cumulative flow diagram
- Lead time et Cycle time
- Happiness index et métriques qualitatives
- Anti-patrons : vanity metrics vs actionable metrics

- **TD/Pratique:**
  - Analyse de métriques réelles (jeu de données fourni) : que conclure ?

Séance 10 : Agilité et Organisation

- Transformation agile d'entreprise : défis et freins
- Le rôle du management dans l'agilité
- Culture d'apprentissage et safe-to-fail
- Agilité à l'échelle (rappel léger)

- **TD/Pratique :**
  - Débat structuré : "L'agilité fonctionne-t-elle dans les grandes entreprises ?"

Séance 11 : Atelier Pratique - Simulation Projet

- Mini-hackathon agile : Les étudiants forment 3-4 équipes Scrum
- Mise en situation complète : création d'un backlog, planning poker, sprint de 3 itérations de 20min chacune avec reviews et rétrospectives
- Livrable : prototype fonctionnel (même basique) ou maquette détaillée

Séance 12 : Examen Final et Synthèse

- 40% questions théoriques (manifeste, rôles Scrum, concepts clés)

- 30% études de cas à analyser
- 30% problèmes pratiques (estimation, priorisation)

- **Synthèse et Q&R:**
  - Retour sur les apprentissages clés
  - Préparation aux contextes professionnels
  - Ressources pour approfondir (certifications PSM, CSM, livres recommandés)

Recommandations pédagogiques pour débutants :

1. **Progressivité** : Commencer par les concepts avant les outils
2. **Expérientiel** : Au moins 40% de pratique (ateliers, simulations)
3. **Contextualisation** : Utiliser des exemples du monde étudiant (applis, projets académiques)
4. **Anti-complexité** : Éviter SAFe et les frameworks lourds en début de cours

SÉANCE 1 : Introduction et Contexte

Objectifs pédagogiques

À la fin de cette séance, l'étudiant sera capable de :

- Expliquer les limites des méthodes traditionnelles (cycle en V)
- Citer les 4 valeurs et au moins 6 des 12 principes du manifeste agile
- Argumenter sur l'intérêt de l'agilité dans un contexte donné

Introduction

- **Accroche** : "Vous arrivez dans une entreprise. On vous dit : 'On fait du cycle en V depuis 20 ans, ça marche parfaitement.' Que répondez-vous ?"
- Sondage rapide : Qui a déjà entendu parler d'agilité ? Dans quel contexte ?

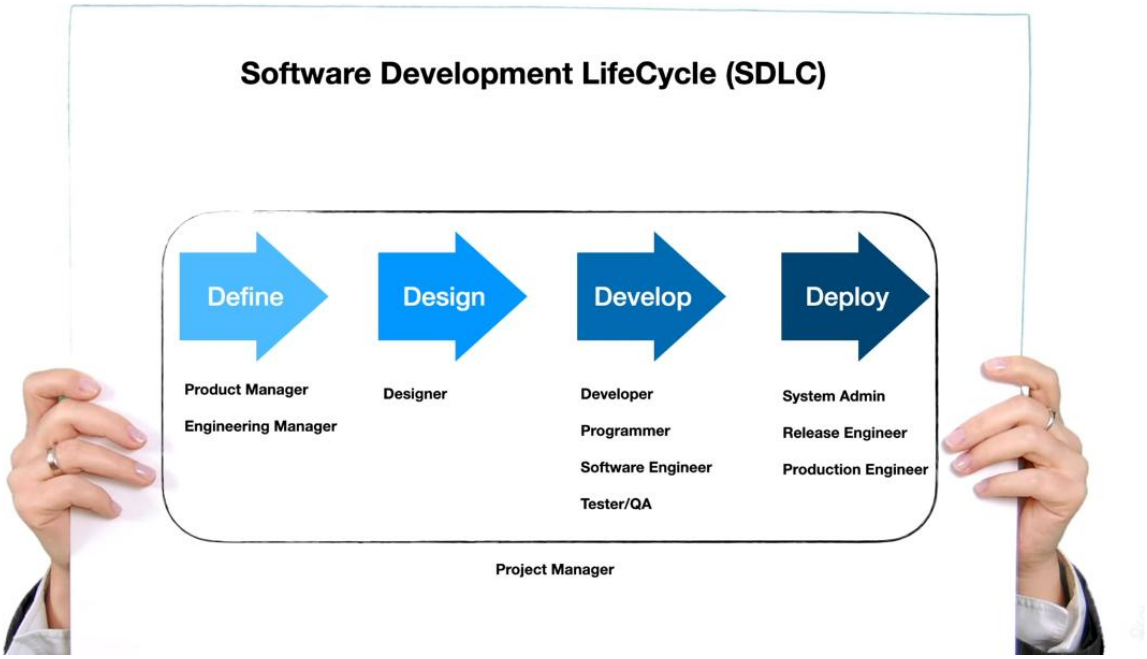
I. Les méthodes traditionnelles et leurs limites

Le modèle du Cycle en V (rappel)

Exigences → Conception → Développement → Tests → Déploiement

↑_____|

- Séquentialité stricte
- Documentation lourde
- Validation tardive par l'utilisateur



Les symptômes de l'échec (données chiffrées)

- Étude Standish Group (Chaos Report) : 70% des projets IT traditionnels échouent partiellement ou totalement
- Coût d'un bug : 100€ en phase d'exigences → 10 000€ en production

Exercice mental : "Imaginez un projet de 12 mois. À quel moment découvrez-vous que vous avez mal compris le besoin client ?"

II. Naissance de l'agilité

Contexte historique

- Années 1990 : émergence de méthodes légères (XP, Scrum, Crystal, FDD...)
- Février 2001 : Snowbird, Utah — 17 experts se réunissent
- Naissance du "Manifeste for Agile Software Development"

Les signataires fondateurs (présentation rapide) :

- Kent Beck (XP), Ken Schwaber & Jeff Sutherland (Scrum), etc.

III. Le Manifeste Agile — Les 4 valeurs

Valeur	On privilégie	plutôt que
1	**Les individus et leurs interactions**	Les processus et les outils
2	**Des logiciels opérationnels**	La documentation exhaustive

Valeur	On privilégie	plutôt que
3	**La collaboration avec les clients**	La négociation contractuelle
4	**L'adaptation au changement**	L'exécution d'un plan

Formulation clé : "Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres. Par ce travail, nous en sommes venus à valoriser..."

Discussion : Pourquoi "plutôt que" et pas "au lieu de" ? → L'agilité n'est pas l'absence de processus ou de documentation, mais un équilibre différent.

IV. Les 12 principes (focus sur 6 essentiels)

#	Principe	Implication concrète
1	Satisfaire le client par des livraisons précoces et continues	Itérations courtes, valeur métier dès la première semaine
2	Accueillir favorablement les changements	Flexibilité contractuelle, pas de "scope freeze"
3	Livrer fréquemment des logiciels opérationnels	Cycle de release : 2 semaines à 2 mois
4	Faire travailler ensemble les acteurs du projet daily	Co-localisation ou outils de collaboration temps réel
5	Simplicité — l'art de minimiser le travail inutile	YAGNI (You Ain't Gonna Need It)
6	Régulièrement, l'équipe réfléchit à devenir plus efficace	Rétrospectives systématiques

Les 6 autres principes : distribution d'une fiche récapitulative pour lecture autonome.

Activité 1 : Quiz interactif

Format : 8 questions à choix multiples, réponses via levée de main ou outil (Mentimeter/Kahoot)

1. *Quelle valeur du manifeste agile correspond à cette situation ?*
   "L'équipe préfère discuter 10 minutes autour d'un tableau blanc plutôt que d'écrire un document de spécifications de 50 pages."
   → Réponse : Individus et interactions > Processus et outils

2. *Vrai ou Faux ?*
   "Le manifeste agile interdit toute documentation."
   → Faux : "La documentation exhaustive" est déconseillée, pas la documentation utile.

3. *Scénario à analyser* :
   Un client demande une modification majeure après 3 mois de développement. En méthode agile, on refuse car le cahier des charges est signé.
   → Faux : L'agilité accueille le changement, même tardif.

Activité 2 : Discussion en sous-groupes

Consigne : "Identifiez 3 situations concrètes (projets étudiants, associatifs, personnels) où une approche agile aurait été préférable à une approche planifiée."

Déroulement :

- 10 min : Réflexion par groupes de 3-4 étudiants
- 10 min : Mise en commun (2-3 groupes présentent)
- 5 min : Synthèse professeur — L'agilité s'applique aussi hors IT (marketing, recherche, événementiel)

Évaluation formative de la séance

- Question de sortie (exit ticket) : "En une phrase, définissez l'agilité pour un néophyte."

SÉANCE 2 : Scrum — Fondamentaux

Objectifs pédagogiques

À la fin de cette séance, l'étudiant sera capable de :

- Décrire la philosophie Scrum (empirisme, itératif-incrémental)
- Identifier et expliquer les 3 rôles Scrum
- Définir les 3 artefacts et leur lien

Introduction

- Rappel séance 1 : L'agilité c'est un état d'esprit, Scrum c'est un cadre (framework)
- Métaphore : "Si l'agilité est la cuisine healthy, Scrum est une recette précise"

I. Scrum

Définition officielle (Scrum Guide 2020) :

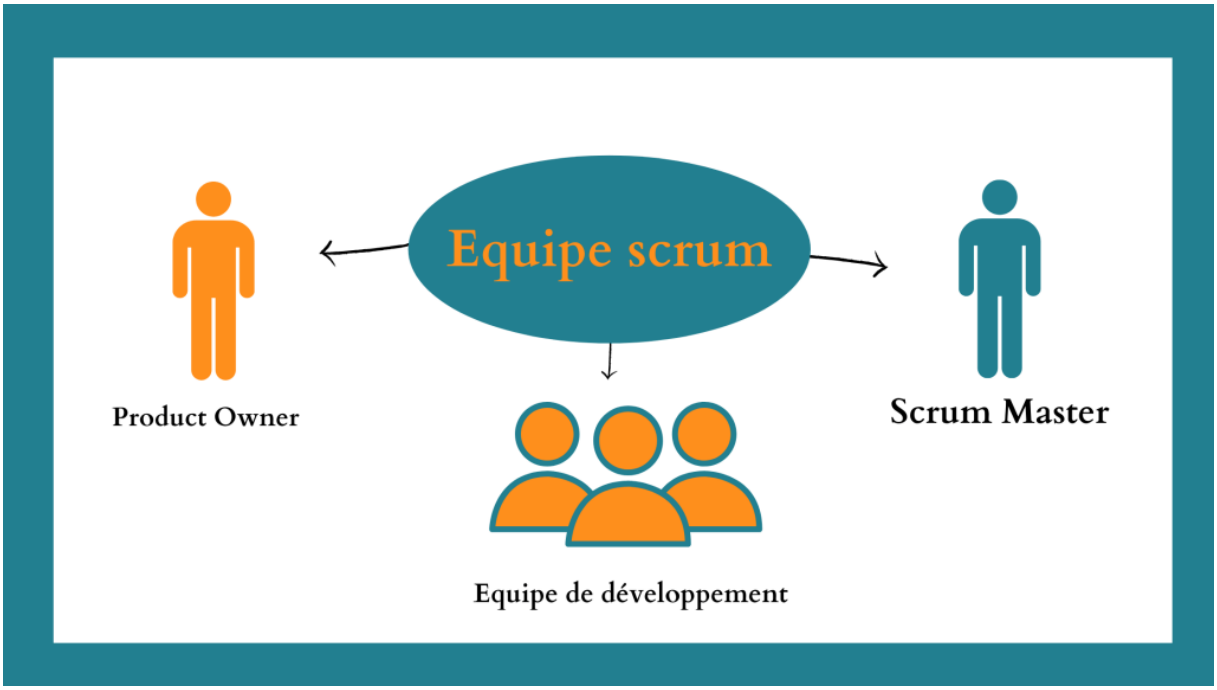
"Un cadre de travail léger qui aide les personnes, les équipes et les organisations à générer de la valeur grâce à des solutions adaptatives pour des problèmes complexes."

Les piliers de Scrum :

1. **Transparence** : Les aspects essentiels du processus sont visibles
2. **Inspection** : Inspection fréquente des artefacts pour détecter les écarts
3. **Adaptation** : Ajustement immédiat si dérive constatée

L'empirisme : La connaissance vient de l'expérience et de la prise de décision basée sur ce qui est connu.

II. Les 3 Rôles Scrum



1. Le Product Owner (PO) — *Le "quoi" → Qui définit les priorités*

- **Responsabilité unique** : Maximiser la valeur du produit
- **Gestion du Product Backlog** : contenu, ordonnancement, visibilité
- **Représentant des parties prenantes** mais pas leur esclave
- **Un seul PO par produit** (pas de comité)

Exemple concret :
Dans une appli de livraison de repas, le PO décide que "paiement en ligne" est prioritaire sur "système de fidélité" car c'est un frein à l'achat immédiat.

2. Le Scrum Master (SM) — Le "comment" du processus → Qui facilite le processus

- **Servant-leader** : sert l'équipe, pas un manager hiérarchique
- **Facilitateur** : anime les cérémonies, élimine les obstacles (impediments)
- **Coach agile** : enseigne Scrum, protège l'équipe des interférences externes
- **Garant du cadre** : fait respecter les règles du jeu

Analogie : Le SM est comme l'arbitre d'un match — il ne joue pas, mais fait respecter les règles pour que le jeu se déroule bien.

3. L'Équipe de Développement (Dev Team) — Le "comment" technique → Qui réalise le travail

- **Auto-organisée** : personne ne dit à l'équipe comment transformer le backlog en incrément
- **Pluridisciplinaire** : toutes les compétences nécessaires sont dans l'équipe
- **Taille** : 3 à 9 personnes (idéalement 5-6)
- **Responsabilité collective** : "Nous avons livré" pas "j'ai fini ma part"

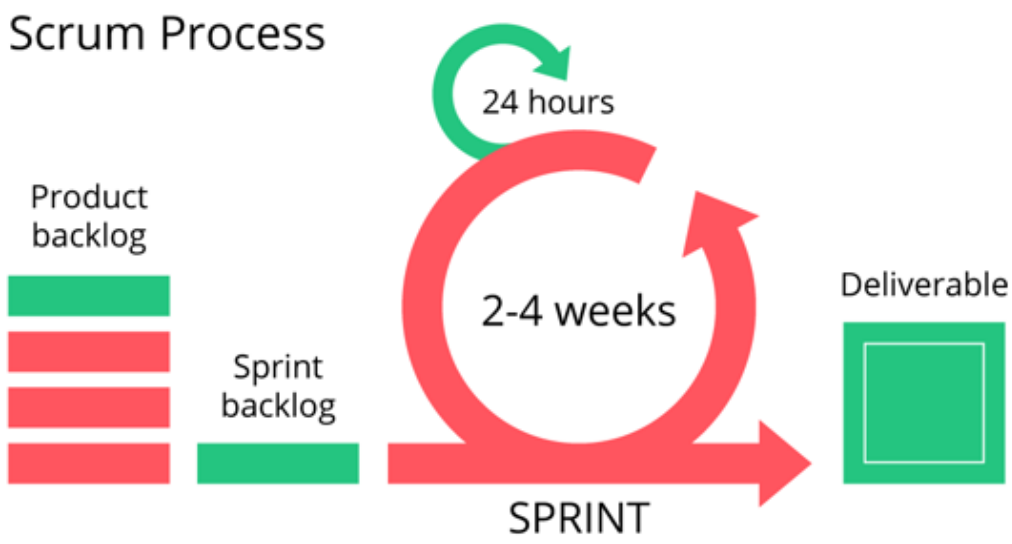
Tableau récapitulatif des rôles :

Rôle	Question clé	Autorité sur
Product Owner	*Quoi construire ?*	Product Backlog, priorisation, ROI
Scrum Master	*Comment bien faire Scrum ?*	Processus, cadence, amélioration continue
Équipe Dev	*Comment construire ?*	Choix techniques, estimation, engagement

Anti-patrons à éviter :

- Le PO qui dicte le "comment" technique
- Le SM qui devient secrétaire de réunion
- Le manager extérieur qui assigne les tâches à l'équipe

III. Les 3 Artefacts Scrum



1. Product Backlog

- **Quoi** : Liste unique et priorisée de tout ce qui pourrait être nécessaire au produit
- **Nature** : Dynamique, jamais complet, évolue constamment
- **Format** : User stories, bugs, tâches techniques, spikes...
- **Propriétaire** : Product Owner (seul à décider de l'ordonnancement)

2. Sprint Backlog

- **Quoi** : Éléments du Product Backlog sélectionnés pour le Sprint + plan pour les réaliser
- **Nature** : Appartient à l'équipe de développement, visible pour tous
- **Évolution** : Peut évoluer durant le Sprint si le travail reste compris dans l'objectif

3. L'Incrément

- **Quoi** : Somme de tous les éléments du backlog terminés durant le Sprint + incréments précédents
- **Condition** : Doit être "Done" (potentiellement livrable)
- **Objectif** : Créer de la valeur, pas juste faire des tâches

Schéma de flux :

Product Backlog → (Sprint Planning) → Sprint Backlog → (Développement) → Incrément "Done"

↑ |

└──────────────────(**Feedback, nouvelles idées**)──────────────────┘

IV. La Definition of Done (DoD)

- Critères communs de qualité définis par l'équipe
- Exemples : code revu, tests passés, documentation à jour, déployé en pré-prod...

Étude de cas : "La Startup FoodTech"

Contexte :
L'équipe "AbdelaliSCRAM" développe une app de livraison de repas étudiants. Elle vient de recruter :

- **Widad** (ex-marketing) : nouvelle PO, connaît bien les étudiants
- **Yassine** (développeur senior) : nommé Scrum Master car "il est organisé"
- **Équipe de 4 devs** : 2 frontend, 2 backend, embauchés récemment

Problématiques à analyser (en groupes de 4) :

1. Widad veut valider chaque ligne de code avant commit. Elle demande aussi aux devs de prioriser les bugs qu'elle identifie en direct sur son téléphone. Quels problèmes voyez-vous ?
2. Yassine continue à coder 50% de son temps et dit aux devs juniors quelle architecture utiliser. Il manque souvent les Daily Scrum car il est en réunion avec le CEO. Analysez.
3. Les devs backend refusent de toucher au frontend car "ce n'est pas leur job". Résultat : les tâches restent bloquées. Que recommandez-vous ?

Déroulement :

- 15 min : Analyse en sous-groupes (fiche distribuée avec les 3 cas)
- 20 min : Débat en plénière, confrontation des solutions
- 10 min : Correction/synthèse professeur avec référence au Scrum Guide

Grille d'analyse attendue :

Cas	Problème identifié	Principe Scrum violé	Solution
1	PO micro-manage le "comment"	Séparation des rôles	PO se concentrer sur le "quoi", faire confiance à l'équipe
2	SM n'est pas disponible + impose solutions	Servant-leadership, auto-organisation	SM à plein temps, coaching pas commandement
3	Silos techniques	Équipe pluridisciplinaire	Former les devs, abolir les silos ou recruter des profils full-stack

SÉANCE 3 : Scrum — Cérémonies et Flux

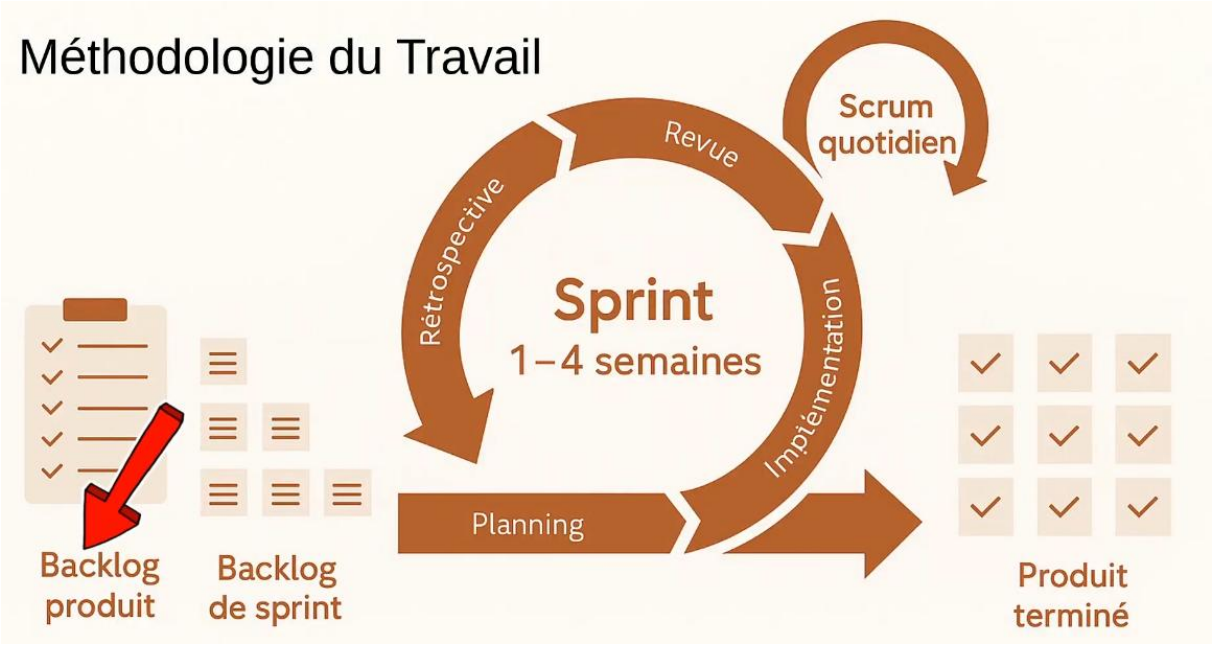
Objectifs pédagogiques

À la fin de cette séance, l'étudiant sera capable de :

- Planifier et animer les 5 événements Scrum
- Expliquer la time-boxing et son importance
- Simuler un Daily Scrum efficace

Introduction

- Rappel : Les artefacts sont statiques, les événements sont dynamiques
- Principe de **time-boxing** : durée fixe maximum, jamais dépassée — "la contrainte crée la créativité"



I. Le Sprint

Définition : Un Sprint est une itération de durée fixe (1 à 4 semaines, idéalement 2 semaines) durant laquelle est créé un incrément "**Done**" utilisable.

Règles fondamentales :

- Durée constante (pas de "Sprint de 10 jours puis 15 jours")
- Pas de modification de l'objectif de Sprint en cours
- Scope négociable avec le PO si nécessaire
- Annulation possible mais rare (décision du PO)

Analogie : Le Sprint est comme une course de vitesse — on ne change pas la distance en cours, mais on peut ajuster sa technique.

II. Sprint Planning

Objectif : Répondre à 3 questions :

1. **Pourquoi ce Sprint est-il utile ?** → Définir l'Objectif de Sprint
2. **Quel sera le travail réalisé ?** → Sélectionner les items du Product Backlog

3. **Comment réaliser ce travail ?** → Créer le Sprint Backlog

Durée : ***2h pour un Sprint de 2 semaines (max 8h pour 1 mois)***

Participants : PO, SM, Équipe Dev (+ stakeholders si besoin pour le "pourquoi")

Sorties :

- Objectif de Sprint clair et motivant
- Sprint Backlog initial
- Prévision de livraison

Exemple d'Objectif de Sprint :
❌ "Faire les user stories US-12, US-15, US-18"
✅ "Permettre aux étudiants de commander un repas en moins de 3 minutes"

III. Daily Scrum

Objectif : Inspection de la progression vers l'Objectif de Sprint + adaptation du Sprint Backlog

Format classique (3 questions par personne) :

1. Qu'ai-je fait hier pour aider l'équipe à atteindre l'objectif ?
2. Que vais-je faire aujourd'hui pour aider l'équipe à atteindre l'objectif ?
3. Quels obstacles risquent d'empêcher l'équipe d'atteindre l'objectif ?

Règles strictes :

- **15 minutes max** (time-boxé)
- Même heure, même lieu
- Toute l'équipe Dev présente (PO et SM peuvent assister mais ne parlent que si nécessaire)
- Ce n'est PAS un reporting au SM !

Focus sur la progression collective, pas sur l'activité individuelle.

IV. Sprint Review

Objectif : Inspection de l'incrément + adaptation du Product Backlog

Format : Démonstration informelle (pas de PowerPoint) de l'incrément "**Done**"

- L'équipe présente ce qui a été fait (et ce qui n'a pas été fait)
- Discussion sur ce qui a bien fonctionné, les problèmes rencontrés
- Collaboration sur ce qui faire ensuite (le PO prend note des feedbacks)

Durée : 1h pour un Sprint de 2 semaines (max 4h pour 1 mois)

Participants : Équipe Scrum + stakeholders clés (clients, utilisateurs, management)

Sortie : Product Backlog mis à jour (nouvelles idées, repriorisation)

V. Sprint Retrospective

Objectif : Amélioration continue du processus et des pratiques

3 questions fondamentales :

1. Qu'est-ce qui a bien fonctionné ? (à conserver)
2. Qu'est-ce qui pourrait être amélioré ? (à changer)
3. Qu'allons-nous mettre en place au prochain Sprint ? (1-2 actions concrètes)

Durée : 45min pour un Sprint de 2 semaines (max 3h pour 1 mois)

Règles de facilitation :

- Toute l'équipe Scrum (pas d'extérieur)
- Lieu sécurisant psychologiquement (pas de jugement)
- Actions concrètes et assignées, pas juste des constats

Techniques alternatives : Speedboat, Starfish, 4L (Liked, Learned, Lacked, Longed for)

Simulation : Daily Scrum

Mise en situation :
Équipe de 5 personnes (rôles assignés : 3 devs, 1 PO, 1 SM) travaillant sur un projet "Appli de réservation de salles de réunion".

Contexte du Sprint :

- Objectif : "Permettre aux étudiants de réserver une salle en 2 clics"
- Jour 3 du Sprint
- Hier : problème technique sur l'API de calendrier

Distribution des fiches personnages (5 min de préparation) :

Personnage	Situation hier	Situation aujourd'hui	Obstacle
Dev 1 (Frontend)	Terminé l'interface de sélection des salles	Intégrer l'API calendrier	L'API ne répond pas ce matin
Dev 2 (Backend)	Investigé le bug API	Continuer l'investigation ou contourner	Besoin de l'aide de Dev 1 pour tester
Dev 3 (Full-stack)	Aidé Dev 1 sur le CSS responsive	Commencer la fonctionnalité "annulation"	Pas clair sur la règle métier d'annulation (24h avant ?)

Personnage	Situation hier	Situation aujourd'hui	Obstacle
PO	Réuni avec les utilisateurs pour valider les maquettes	Disponible pour clarifier	-
SM	Résolu le problème d'accès au serveur de test	Faciliter la Daily	-

Déroulement de la simulation:

- 15 min : Daily Scrum réelle (timer visible : 15 min max)
- 10 min : Débrief collectif
  - ✓ L'objectif de Sprint était-il au centre ?
  - ✓ Le SM est-il intervenu utilement ?
  - ✓ Des actions ont-elles émergé ?
- 5 min : Retour professeur — pièges classiques à éviter

Pièges observés et corrigés :

- Dérive vers résolution technique → "On prend ça après la Daily"
- Reporting au SM → "Adresse-toi à l'équipe, pas à moi"
- Oubli de l'objectif de Sprint → "Rappel : on vise la réservation en 2 clics, l'API bloque ça"

Variante : Rejouer avec un "perturbateur" (étudiant qui parle trop, qui arrive en retard, qui sort son téléphone) pour entraîner le SM à faciliter.

Fiches récapitulatives à distribuer : Fiche A4 "Les 5 Événements Scrum" :

Événement	Fréquence	Durée (Sprint 2 sem.)	Objectif clé
Sprint	Tout le Sprint	2 semaines	Créer un incrément de valeur
Sprint Planning	Début du Sprint	2h	Planifier le Sprint

Événement	Fréquence	Durée (Sprint 2 sem.)	Objectif clé
Daily Scrum	Quotidien	15 min	Synchroniser l'équipe
Sprint Review	Fin du Sprint	1h	Démontrer et collecter feedback
Sprint Retrospective	Fin du Sprint	45 min	S'améliorer pour le prochain

SÉANCE 4 : User Stories et Product Backlog

Objectifs pédagogiques

À la fin de cette séance, l'étudiant sera capable de :

- Rédiger des user stories selon le format standard
- Définir des critères d'acceptation exploitables
- Prioriser un backlog avec des techniques adaptées

Introduction

- Rappel : Le Product Backlog contient des "items" — mais comment les formuler ?
- Transition : Du besoin vague à l'item actionnable

I. Le concept de User Story

Définition (Mike Cohn) :

"Une user story est une description courte et simple d'une fonctionnalité racontée du point de vue de la personne qui désire cette nouvelle capacité, généralement un utilisateur ou un client du système."

Format standard (Connextra) :

```
En tant que [rôle]
Je veux [fonctionnalité/besoin]
Afin de [bénéfice/valeur]
```

Exemples comparés :

Mauvaise formulation	Bonne formulation
"Système de connexion"	"En tant qu'étudiant, je veux me connecter avec mon email universitaire afin d'accéder à mes cours sans créer de nouveau compte"
"Base de données utilisateurs"	"En tant qu'administrateur, je veux consulter la liste des utilisateurs inactifs depuis 6 mois afin de nettoyer les comptes obsolètes"

Les 3 C de Ron Jeffries :

1. **Card** (Carte) : Physique ou digitale, limite la taille
2. **Conversation** : Le support d'un dialogue entre PO et équipe
3. **Confirmation** : Les critères d'acceptation qui valident la story

INVEST — Les 6 qualités d'une bonne user story :

Lettre	Critère	Explication
I	Independent	Peut être développée séparément (souvent impossible à 100%, mais viser la dépendance minimale)
N	Negotiable	Négociable entre PO et équipe, pas un contrat figé
V	Valuable	Apporte de la valeur métier, pas juste technique
E	Estimable	L'équipe peut l'estimer (pas trop vague)
S	Small	Assez petite pour tenir dans un Sprint
T	Testable	On peut vérifier objectivement qu'elle est réalisée

II. Les Critères d'Acceptation

Définition : Conditions précises qui déterminent si la story est "Done" du point de vue métier.

Format Given-When-Then (Gherkin) :

```
Étant donné [contexte initial]
Quand [action réalisée]
Alors [résultat attendu]
```

Exemple complet :

User Story : "En tant qu'étudiant, je veux réserver une salle afin de préparer mon exposé en groupe"

Critères d'acceptation :

- Étant donné que je suis connecté, quand je sélectionne une salle disponible, alors elle est marquée "réservée" à mon nom
- Étant donné que je sélectionne une salle déjà réservée, quand je clique sur "réserver", alors un message m'indique l'indisponibilité
- Étant donné que ma réservation est confirmée, quand l'heure approche, alors je reçois un rappel par notification

Différence clé :

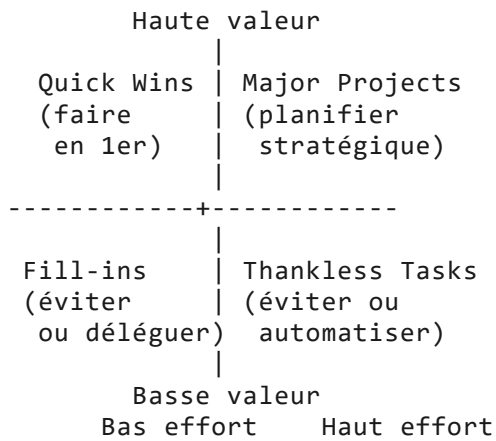
- User Story = **Quoi** et **Pourquoi** (besoin métier)
- Critères d'acceptation = **Comment vérifier** (scénarios de test)

III. Techniques de priorisation

1. MoSCoW :

- **M**ust have : Indispensable pour le MVP
- **S**hould have : Important mais pas critique
- **C**ould have : Souhaitable si temps disponible
- **W**on't have (this time) : Explicitement reporté

2. Valeur métier vs Effort (Matrice 2x2) :



3. WSJF (Weighted Shortest Job First) — Introduction légère :

- Coût du délai / Durée = Priorité
- Utilisé quand les ressources sont limitées et les opportunités concurrentes

4. Kano Model (mention rapide) :

- Basiques (attendus), Performeurs (plus c'est mieux), Excitateurs (surprennent)

IV. Spikes et Technical Stories

- **Spike** : Recherche technique de durée limitée pour réduire l'incertitude
- **Technical Story** : Tâche technique sans valeur utilisateur directe mais nécessaire (ex: "migrer la base de données")

PARTIE TD/PRATIQUE

Atelier : Construction d'un Product Backlog (45 min)
Contexte : Application "CampusBuddy" — aide aux étudiants pour la vie universitaire.

Phase 1 : Brainstorming (15 min) En groupes de 4, générer 15-20 idées de fonctionnalités sur des post-its (physiques ou digitaux via Miro/FigJam).

Phase 2 : Rédaction (15 min) Transformer 5 idées en user stories INVEST + critères Given-When-Then.

Exemple de correction collective :

Idée brute : "Système de messagerie entre étudiants"
User Story : "En tant qu'étudiant en groupe de projet, je veux créer un canal de discussion dédié à mon groupe afin de centraliser nos échanges sans dépendre de WhatsApp"
Critères :

- GWT : Connexion → Création canal → Canal visible dans ma liste
- GWT : 5 membres max par canal (règle métier)
- GWT : Notification push activable/désactivable par canal

Phase 3 : Priorisation (15 min) Classer les 5 stories avec MoSCoW, justifier les choix en 2 phrases.

Livrable : Chaque groupe remet une fiche avec :

- 5 user stories formatées
- Critères d'acceptation associés
- Priorisation MoSCoW justifiée

SÉANCE 5 : Estimation et Planification Agile

Objectifs pédagogiques

À la fin de cette séance, l'étudiant sera capable de :

- Expliquer pourquoi l'estimation relative l'emporte sur l'estimation absolue
- Utiliser le Planning Poker et interpréter la divergence d'estimation
- Calculer et utiliser la velocity pour la prédictibilité

Introduction

- **Paradoxe de l'estimation** : Plus on est précis, plus on est probablement faux
- **Loi de Hofstadter** : "Toujours plus long que prévu, même en tenant compte de la loi de Hofstadter"
- **Principe agile** : On estime l'effort, pas le temps — car le temps dépend des individus

I. Pourquoi l'estimation relative ?

Problème de l'estimation en heures/jours :

- Dépend de l'expérience du développeur
- Crée de la pression et du "padding" (sur-estimation défensive)
- Faux sentiment de précision ("3.5 jours" vs "3 ou 8 jours")

Principe de l'estimation relative :

"Cette story est-elle plus grosse, plus petite, ou de taille similaire à celle-ci ?"

Analogie : Comparer la taille de deux bâtiments — on ne mesure pas en mètres, on dit "celui-ci est 2 fois plus grand".

II. Les Story Points

Définition : Unité abstraite mesurant la complexité, l'effort et l'incertitude combinés.

Échelle de Fibonacci modifiée : 1, 2, 3, 5, 8, 13, 21, 40, 100

- Les écarts croissants reflètent l'incertitude croissante
- Pas de 4, 6, 7 — car la différence entre 5 et 8 est moins significative que celle entre 1 et 2

Facteurs pris en compte :

- Complexité technique
- Volume de travail
- Incertitude / risque

Ce qui NE doit PAS influencer :

- Qui fait le travail (l'équipe estime, pas l'individu)
- Urgence artificielle ("le client veut ça vite")

Référence (baseline) :

- Choisir une story "1 point" connue et bien comprise par l'équipe
- Toutes les autres stories se comparent à cette référence

III. Le Planning Poker (15 min)

Matériel : Cartes avec les valeurs de la suite de Fibonacci

Déroulement :

1. Le Product Owner lit la user story et les critères d'acceptation
2. L'équipe pose des questions de clarification (2-3 min max)
3. Chaque membre choisit secrètement une carte

1. **Révélation simultanée** (crucial pour éviter l'ancrage)
2. Discussion si divergence significative :
   - Le plus haut explique ses risques perçus
   - Le plus bas explique sa simplicité perçue
3. Nouveau vote si nécessaire (max 3 tours)

Règles :

- Pas de négociation avant révélation
- Le PO ne vote pas (il n'est pas développeur)
- Le SM facilite, ne vote pas (sauf s'il est aussi dev)

Interprétation de la divergence :

- Écart 1-2 niveaux → Moyenne ou consensus rapide
- Écart 3+ niveaux → Signe d'incertitude à investiguer (spike ?)

IV. Velocity et Prédictibilité

Définition : Nombre moyen de story points réalisés par Sprint.

Calcul :

```
Velocity = Σ(story points des items "Done") / nombre de Sprints
```

Utilisation :

- Prévision : "Avec une velocity de 23 points, nous pouvons nous engager sur 20-25 points"
- **Jamais** pour évaluer la performance individuelle
- **Jamais** pour comparer deux équipes

Le cone of uncertainty :

- Sprint 1-2 : Velocity instable (±50%)
- Sprint 3-6 : Stabilisation (±20%)
- Sprint 6+ : Prédictibilité acceptable

PARTIE TD/PRATIQUE

Atelier Planning Poker
Contexte : Suite du projet "CampusBuddy" — estimer le backlog créé séance 4.

Préparation (5 min) :

- Baseline établie : "Se connecter avec email universitaire" = 3 points
- Distribution des cartes (physiques ou application en ligne)

Déroulement (35 min) :

- 5 stories à estimer (sélectionnées par le professeur pour variété de complexité)
- Pour chaque story : lecture, questions, vote, discussion si écart > 2 niveaux
- Le professeur joue le PO, un étudiant le SM

Stories proposées :

1. "S'inscrire à un événement campus" (simple, probablement 3-5)
2. "Payer sa cotisation associative en ligne" (complexe, intégration bancaire, probablement 13-21)
3. "Consulter l'emploi du temps" (moyen, probablement 5-8)
4. "Recevoir des notifications personnalisées" (moyen+, probablement 8-13)
5. "Partager son emploi du temps avec un groupe" (complexe, synchronisation, probablement 13-21)

Débrief (5 min) :

- Comparer les estimations entre groupes

- Discuter : "Pourquoi l'écart sur la story 2 ? Qu'est-ce qui créait l'incertitude ?"