# Présentation XP

## Introduction

Les piliers de l'eXtreme Programming (XP) reposent sur des mécanismes clés pour :

- Optimiser l'efficacité des équipes (pair programming, intégration continue, etc.),
- Raccourcir les boucles de feedback (livraisons fréquentes, tests automatisés, rétrospectives),
- Fluidifier les releases (déploiement continu, petites itérations incrémentales),
- Renforcer la collaboration client (user stories, planification adaptative, démonstrations régulières).

La maîtrise de XP passe par une compréhension hiérarchisée :

- Ses Valeurs (communication, simplicité, feedback, courage, respect) → fondations culturelles.
- Ses Principes (ex : "Embrasser le changement", "Livrer de la valeur rapidement") → cadrage stratégique pour choisir/adapter les pratiques.
- Ses Pratiques (TDD, refactoring, CI/CD, velocity tracking, etc.) → outils concrets pour implémenter les principes.

Adaptation contextuelle :

XP n'est pas un framework rigide. Certaines pratiques (ex : pair programming) peuvent être ajustées ou combinées (avec Scrum/Kanban) selon :

- La taille de l'équipe (startup vs. scale-up),
- La criticité du projet (MVP vs. système embarqué),
- La maturité technique (legacy code vs. greenfield).

XP Agile DevOps TDD Clean code

### Valeurs XP

Les **valeurs XP (eXtreme Programming)** définissent la culture et les comportements clés pour des équipes **agiles et techniques**.

#### Tableau des Valeurs XP

| Valeur | Définition Technique |
|---|---|
| **Communication** | **Échanges structurés** pour : |
| | - *Knowledge sharing* : **mob programming**, docs collaboratives (ex : Confluence/Notion). |
| | - *Résolution de problèmes* : **stand-ups techniques**, canaux dédiés (Slack/Teams). |
| | - *Coordination* : **planning poker**, raffinement de backlog. |
| **Simplicité** | **Approche minimaliste** : |
| | - *YAGNI* : **"You Aren't Gonna Need It"** → pas de sur-ingénierie. |
| | - *DTSTTCPW* : **"Do The Simplest Thing That Could Possibly Work"**. |
| | - *Refactoring* : **incrémental** (ex : *boy scout rule*). |
| **Feedback** | **Boucles courtes et automatisées** : |
| | - *Tests* : **TDD/BDD** (feedback en ms). |
| | - *Démos* : **Sprint reviews** (validation client en jours). |
| | - *Rétros* : **Amélioration continue** des processus. |

| Valeur | Définition Technique |
|---|---|
| **Courage** | **Décisions techniques audacieuses** : |
| | - Remettre en question les choix (ex : “Ce framework est-il adapté ?”). |
| | - *Fail fast* : **Spikes** pour valider des hypothèses avant le dev. |
| | - *Blameless culture* : **Postmortems** sans jugement. |
| **Respect** | **Collaboration bienveillante** : |
| | - *Pair programming* : **Montée en compétence** (juniors/seniors). |
| | - *Code reviews* : **Focus sur l’amélioration**, pas la critique. |
| | - *Retros actionables* : **Écoute active** des feedbacks. |

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
**http://slamwiki2.kobject.net/eadl/bloc3/xp/chap1?rev=1763862043**

Last update: **2025/11/23 02:40**