

MGL7250 – Processus de développement Agile  
eXtreme Programming  
Jacques Berger

# Objectifs

Présenter la méthode eXtreme Programming

# Prérequis

Aucun

# XP

eXtreme Programming

Méthode conçue par Kent Beck en 1999

# XP

Un ensemble de valeurs, de principes et de pratiques pour améliorer la qualité des logiciels et la productivité des développeurs

Objectif : Développer des logiciels remarquables

# XP

XP est une méthode qu'on peut adopter petit-à-petit, selon nos besoins et selon la maturité de l'équipe (ou l'entreprise)

Les pratiques sont bonnes seules mais sont optimales lorsqu'utilisées toutes ensemble

# XP

XP pousse à l'extrême des pratiques simples et efficaces

Les pratiques de XP favorisent l'adaptation aux changements

# Valeurs

## La communication

Une équipe doit communiquer beaucoup. Un développeur moyen qui communique bien vaut mieux qu'un super-développeur solitaire. La communication règle beaucoup de problèmes humains.



# Valeurs

## La simplicité

On préconise les solutions simples. On essaie toujours la solution qui semble la plus simple.

# Valeurs

## Le feedback

On veut du feedback le plus tôt possible, le plus souvent possible, sur tout (logiciel, code, documentation, etc.)

# Valeurs

## Courage

Dire les choses comme elles sont; écarter les solutions qui ne fonctionnent pas; rechercher des solutions plus simples; rechercher du feedback significatif.

# Valeurs

## Le respect

Respecter son travail et le travail des autres; rien ne peut sauver un projet si un membre de l'équipe s'en fout.

# Principes

## Humanité

Les développeurs sont des personnes. Il faut les traiter comme des personnes. Leurs besoins : sécurité; accomplissement; appartenance; croissance; intimité.

# Principes

## Économique

Ça coûte de l'argent. Poser des gestes qui ont de la valeur; rencontrer les objectifs de l'entreprise.  
Produire de la valeur pour l'entreprise.

# Principes

Bénéfice mutuel

Chaque activité doit profiter à toutes les parties prenantes.

# Principes

## Auto-similarité

Lorsqu'une solution fonctionne, on peut l'utiliser dans un nouveau contexte ou à un différent niveau.



# Principes

## Amélioration

Rien n'est parfait mais on peut perfectionner nos processus, notre code, notre design, nos méthodes de travail, etc.

# Principes

## Diversité

La diversité des compétences dans une équipe est une force. Plusieurs perspectives sur un même problème permet de trouver la solution la plus simple.

# Principes

## Réflexion

Faire son travail, mais également réfléchir sur le fonctionnement et les échecs. Ne pas cacher ses erreurs mais apprendre d'elles.

# Principes

## Flux

Un flux continu d'activités plutôt que des phases distinctes.

# Principes

## Opportunité

Les problèmes sont des opportunités pour améliorer le logiciel.

# Principes

## Redondance

Une pratique pour améliorer la qualité, c'est bien.  
Plusieurs pratiques pour améliorer la qualité, c'est beaucoup mieux.

# Principes

## L'échec

Ne pas avoir peur d'échouer. S'il y a 4 solutions possibles, on essaie les 4. Si les 4 échouent, l'apprentissage acquis en chemin nous aidera à trouver une 5ème solution.

# Principes

## Qualité

On ne peut pas diminuer la qualité. Un projet n'ira pas plus vite si on sacrifie la qualité. On ne négocie jamais sur la qualité.



# Principes

## Petits pas de bébé

Faire des petits changements à la fois; des petites intégrations; des petites mises à jour, etc. Éviter les déploiements de type «Big Bang»

# Pratiques

Les pratiques primaires

La base; les premières pratiques à adopter

# Pratiques

S'asseoir ensemble

L'équipe doit travailler ensemble; aire ouverte; la proximité favorise la communication.

# Pratiques

## L'équipe complète

L'équipe contient toutes les compétences et perspectives nécessaires pour compléter le projet.  
Être engagé dans l'équipe; sentiment d'appartenance.

# Pratiques

## Espace de travail informatif

Avoir une idée de l'avancement du projet ou du fonctionnement de l'équipe en quelques secondes, uniquement en regardant l'espace de travail de l'équipe. Ex : tableaux et post-its

# Pratiques

## Travail sous tension

Maintenir une cadence raisonnable; ne pas brûler les membres de l'équipe; heures de travail limitées. Un développeur travaille mieux lorsqu'il est préparé et bien reposé. Bien gérer le temps productif.

# Pratiques

## Programmer en binôme

2 développeurs sur une machine; deux têtes valent mieux qu'une. Une personne code, l'autre observe l'ensemble (design, erreurs, etc.)

# Pratiques

## Scénarios (stories)

Description d'une fonctionnalité visible par l'utilisateur final. On élabore les scénarios uniquement lorsque c'est nécessaire. On choisit les scénarios à implémenter.



# Pratiques

## Cycle hebdomadaire

Planifier le travail une semaine à la fois. Une réunion au début de la semaine où l'on discute : de la progression du dernier cycle; le client choisit les scénarios à implémenter durant la semaine; on divise les scénarios en tâches.

# Pratiques

## Cycle trimestriel

On planifie le travail un trimestre à la fois.  
Identifier les goulots d'étranglement; planifier les thèmes du trimestre; choisir des scénarios reliés au thème pour le trimestre.

# Pratiques

## Relâchement

Inclure des tâches mineures qu'on peut facilement laisser tomber en cas de retard. L'idée est d'avoir une certaine marge de manoeuvre en cas de problème. On peut toujours rajouter des scénarios et livrer plus que prévu au client si tout va bien.

# Pratiques

Construction en 10 minutes

Automatiser le build. Son exécution ne devrait pas prendre plus de 10 minutes.

# Pratiques

## L'intégration continue

Plusieurs commits par jour pour chaque développeur. On intègre le travail de tous, au fur et à mesure.

# Pratiques

## Les tests d'abord

Rédiger les tests unitaires avant d'écrire le code.  
Écrire le test, écrire le code, refactoring.

# Pratiques

## Conception incrémentale

Investir au quotidien dans le design. Effectuer fréquemment du refactoring.

# Pratiques

## Pratiques corollaires

D'autres pratiques de XP mais qu'il vaut mieux attendre que les pratiques primaires soient bien implantées avant de s'y lancer.



# Pratiques

## Implication réelle du client

Faire participer le client et ses utilisateurs dans le développement. Avoir le client dans son équipe ou travailler chez le client.

# Pratiques

## Déploiement incrémental

Ajouter de nouvelles fonctionnalités à chaque déploiement.

# Pratiques

## Continuité de l'équipe

Lorsqu'une équipe développe une synergie, on préserve la composition de l'équipe d'un projet à l'autre. On ne change pas une équipe gagnante.

# Pratiques

## Réduire les équipes

Lorsque l'équipe gagne en productivité, il est possible de faire la même quantité de travail avec une personne de moins dans l'équipe.

# Pratiques

## L'analyse des causes des défauts

Pour chaque problème, on cherche la cause.  
Ensuite, on cherche la cause de la cause, et ainsi de suite.

# Pratiques

Code partagé (sans propriétaire)

Tous les développeurs peuvent modifier le code source, sans devoir demander la permission.

# Pratiques

Ne maintenir que le code et les tests

Tout le reste ne vaut pas la peine d'être maintenu

# Pratiques

Une seule base de code

Placer plusieurs projets dans le même dépôt  
introduit des opportunités de réutilisation de code



# Pratiques

## Déploiement quotidien

Être en mesure de déployer une nouvelle version de l'application chaque fois. Peut-être même plusieurs fois par jour.

# Pratiques

## Pay-per-use

Faire payer l'utilisateur à l'utilisation plutôt que l'achat d'une licence.

# Changement

Introduire XP dans une entreprise peut entraîner beaucoup de changement; ceci peut être difficile

Les développeurs devront également s'y adapter et pourraient résister à certaines pratiques

Ne brûlez pas d'étapes

# Objectifs

Nous voulons obtenir :

La qualité

La productivité

Ne jamais sacrifier la qualité pour obtenir la productivité

Visez toujours la qualité

# Plus loin...

Extreme Programming Explained : Embrace Change, 2nd edition  
Kent Beck, Cynthia Andres  
Addison-Wesley, 2005

Extreme programming  
<http://www.extremeprogramming.org/>