Extreme Programming

Baptiste Darnala

Université de Montpellier

11 mai 2017

Outline

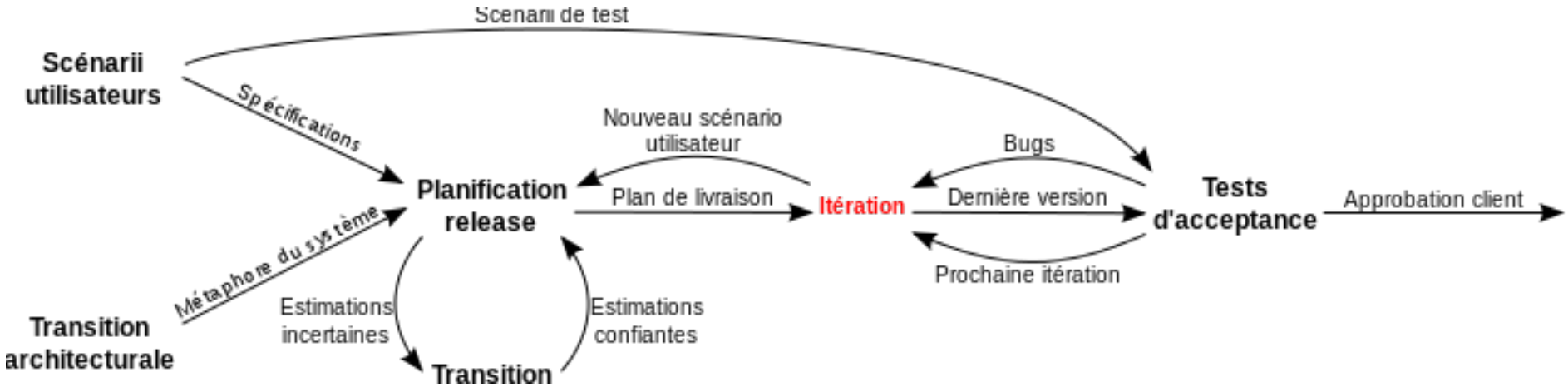
Introduction

L'extreme programming a été inventée par Kent Beck, Ward Cunningham, Ron Jeffries et Palleja Xavier.
Elle fut inventé pendant qu'ils travaillaient sur un projet de calcul de rémunérations chez Chrysler.
En 1999, Kent Beck sort le livre Extreme Programming Explained.

Explication Rapide

L'Extreme Programming (XP) est une méthode agile de travail qui est basé sur du travail en binôme, une présence continue du client et des périodes de travail courtes entre chaque test.

Cycle de développement



Valeurs

L'Extreme programming repose sur 5 valeurs fondamentales :

- Communication
- Simplicité
- Feedback
- Courage
- Respect

Communication

C'est le moyen fondamental pour éviter les problèmes. Les pratiques que préconise l'XP imposent une communication intense. Les tests, la programmation en binôme et le jeu du planning obligent les développeurs, les décideurs et les clients à communiquer.

Simplicité

La façon la plus simple d'arriver au résultat est la meilleure. Une application simple sera plus facile à faire évoluer.

Feedback

Le retour d'information est primordial pour le programmeur et le client. Les tests unitaires indiquent si le code fonctionne et les tests fonctionnels donnent l'avancement du projet. Les livraisons fréquentes permettent de tester les fonctionnalités rapidement.

Courage

Certains changements demandent beaucoup de courage. En effet, il faut parfois changer l'architecture d'un projet, jeter du code pour en produire un meilleur ou essayer une nouvelle technique ce qui peut être assez déstabilisant pour le développeur.

Respect

Cette valeur inclut le respect pour les autres, ainsi que le respect de soi. Les programmeurs ne devraient jamais valider les modifications qui cassent la compilation, qui font échouer les tests unitaires existants ou qui retardent le travail de leurs pairs. Les membres respectent leur propre travail en cherchant toujours la qualité et la meilleure conception pour la solution.

Pratiques

Client sur site

Un représentant du client doit être présent pendant toute la durée du projet et doit avoir les connaissances de l'utilisateur final et avoir une vision globale du résultat final.

Jeu de planning

Le client crée des scénarios pour les fonctionnalités qu'il souhaite obtenir. L'équipe évalue le temps nécessaire pour les mettre en œuvre. Le client sélectionne ensuite les scénarios en fonction des priorités et du temps disponible.

Intégration continue

Lorsqu'une tâche est terminée, les modifications sont immédiatement intégrées dans le produit complet.

Pratiques

Petites livraisons

Les livraisons doivent être les plus fréquentes possible. L'intégration continue et les tests réduisent considérablement le coût de livraison.

Rythme soutenable

L'équipe ne fait pas d'heures supplémentaires. Si le cas se présente, il faut revoir le planning.

Tests fonctionnels

À partir des scénarios définis par le client, l'équipe crée des procédures de test qui permettent de vérifier l'avancement du développement. Si tous les tests fonctionnels passent, l'itération est terminée

Pratiques

Test unitaire

Avant de mettre en œuvre une fonctionnalité, le développeur écrit un test qui vérifiera que son programme se comporte comme prévu. Ce test sera conservé jusqu'à la fin du projet, tant que la fonctionnalité est requise.

Conception simple

La conception doit être simple et direct : chaque idée doit être exprimée clairement, tout doit être écrit une seule fois et le nombre de classes, de méthodes et de lignes de code doit être minimal

Utilisation de métaphores

On utilise des métaphores et des analogies pour décrire le système et son fonctionnement

Pratiques

Refactoring

Amélioration régulière de la qualité du code sans en modifier le comportement. On retravaille le code pour repartir sur de meilleures bases tout en gardant les mêmes fonctionnalités.

Appropriation collective du code

L'équipe est collectivement responsable de l'application. Chaque développeur peut faire des modifications dans toutes les portions du code, même celles qu'il n'a pas écrites.

Convention de nommage

Puisque tous les développeurs interviennent sur tout le code, il est indispensable d'établir et de respecter des normes de nommage pour les variables, méthodes, objets, classes, fichiers, etc.

Pratiques

Programmation en binôme

La programmation se fait par deux. Le premier appelé driver (ou pilote) tient le clavier. C'est lui qui va travailler sur la portion de code à écrire. Le second appelé partner (ou copilote) est là pour l'aider en suggérant de nouvelles possibilités ou en décelant d'éventuels problèmes. Les développeurs changent fréquemment de partenaire ce qui permet d'améliorer la connaissance collective de l'application et d'améliorer la communication au sein de l'équipe.

Rôles

Pendant le développement, chaque personne doit avoir un rôle et doit suivre les directives que lui confère son rôle. Il existe 6 rôles différent :

- Programmeur
- Client
- Testeur
- Tracker
- Manager
- Coach

Programmeur

Le programmeur a un rôle central, il est à la fois codeur, testeur, concepteur et analyste. La rigueur et la communication sont des qualités nécessaires car le travail en binôme et les sessions de programmation courtes sont une partie essentielles de l'XP.

Client

Le client n'est pas forcément le client en lui-même mais peut-être un représentant. Il doit pouvoir apporter une description des fonctionnalités attendues. Il est là pour vérifier que le programme fonctionne comme attendu et donner une ordre des fonctionnalités attendus.

Testeur

Le testeur définit et automatise les tests, il conseil le client sur les tests de fonctionnalités. Il doit savoir maîtriser un outillage de tests et communiquer autant avec le client que avec l'équipe de programmation.

Tracker

Le Tracker a pour mission de suivre les tâches en cours d'itération, interroger les programmeurs sur le déroulement du projet mais il ne doit jamais les mettre sous pression. Il doit aussi être capable de détecter les problèmes avant qu'il soit trop tard.

Manager

Le manager est le supérieur hiérarchique des programmeurs. Il a pour responsabilités de s'occuper de l'équipe, de demander des comptes.

Coach

Le coach est là pour vérifier que tout les rôles sont respecté. Il a pour but que l'équipe soit autonome et que chaque programmeur soit en amélioration constante. Les qualités demandés sont :

- Expert de la méthode XP
- Expert technique
- Programmeur chevronné
- Architecte
- Pédagogue
- Sang-froid

Compatibilités des rôles

	Programmateur	Client	Testeur	Tracker	Manager	Coach
Programmateur		✗	!	!	✗	!
Client	✗		✓	✗	✗	✗
Testeur	!	✓		✗	✗	✗
Tracker	!	✗	✗		!	!
Manager	✗	✗	✗	!		
Coach	!	✗	✗	!	✗	

« ✓ » bonne combinaison
« ! » envisageable mais risquée
« ✗ » mauvaise combinaison

Contextes défavorables

Une telle méthodes agiles ne peut pas être applicables partout, il existe des contextes qui sont défavorables a sa mise en place :

Un blocage culturel

Si le client ou les développeurs ont l'habitude de fonctionner autrement.

Une trop grande équipe

Une équipe de 20 personnes ou plus car la communication devient de plus en plus difficiles quand le groupe est trop grand

Un aménagement pas adapté

L'aménagement du bureau doit permettre une communication facile, c'est pourquoi l'open-space est fortement recommandé.

Contextes défavorables

respect d'une discipline collective

La programmation en binôme est une pratique importante dans cette méthode de travail. Une discipline doit être mis en place pour pouvoir travailler efficacement et certaine personnes peuvent être réfractaire à une telle discipline.

Bibliographie

Wikipedia
https ://fr.wikipedia.org/wiki/Extreme_programming

Cours sur l'XP
http ://www.emse.fr/ picard/cours/2A/gp/GP-ExtremeProgramming.pdf