eXtreme Programming

Emmanuel CHENU
emmanuel.chenu@fr.thalesgroup.com

Cet article répond à des questions concernant l'eXtreme Programming (XP) fréquemment posées dans le cadre professionnel et universitaire. Qu'est-ce que l'eXtreme Programming ? Qu'y a t-il d'extrême ? Où est l'originalité ? Comment s'articule un développement en mode XP ? Comment se mettre à la méthode ? Est-ce applicable pour de grandes équipes et le développement multisite ? Est-ce un retour en arrière ? XP est-il sectaire?
Cet article s'adresse à des développeurs qui ne pratiquent pas XP ou qui n'ont pas connaissance de la seconde édition.

Qu'est-ce que l'eXtreme Programming?

L'eXtreme Programming (XP) est une méthode de développement logiciel mise au point par Kent Beck, Ward Cunningham et Ron Jeffries à la fin des années 90.
Elle recherche l'efficacité maximale en concentrant l'effort de travail sur l'objectif de développer vite et juste. Par "développer juste", il s'agit de bien développer le bon logiciel.
La démarche est légère, pragmatique, disciplinée, empirique et adaptative. XP fait partie de la famille des méthodes agiles de développement logiciel.

Pourquoi un (autre) article sur XP?

XP est un phénomène à noter. Les praticiens d'XP affirment souvent que la méthode a révolutionné leur manière de travailler et qu'ils ne souhaitent plus revenir en arrière. De plus, XP leur permet de prendre du recul en tenant compte des aspects humains du développement logiciel. Aussi, de nombreux praticiens affirment avoir (re)trouvé le plaisir de développer.
Enfin, la plupart des livres, articles et sites Internet font référence à la première édition de XP. C'est pourquoi cet article se base sur la seconde édition, décrite par Kent Beck en 2005.

Pourquoi une seconde édition?

La seconde édition est surtout une reformulation complète de la première. Les pratiques sont découpées plus finement et complétées par des nouvelles. Aussi, l'auteur justifie la discipline en décrivant la philosophie et les motivations qui sont les racines de la démarche.

Pourquoi "eXtreme" ?

Afin de viser une efficacité maximale en développement, XP identifie un jeu de bonnes pratiques, les combine et pousse leur application à l'extrême.
Par exemple,

- La revue de code est une bonne pratique, elle est donc menée en continue;
- Les tests sont primordiaux, ils sont donc écrits systématiquement avant d'implémenter et rejoués en continue;
- La conception est importante, elle est donc menée et remaniée tout au long du projet;
- La simplicité permet d'avancer plus vite, ainsi la solution viable la plus simple est toujours retenue;
- L'intégration des modifications est cruciale, elle est donc réalisée plusieurs fois par jour;
- Les besoins évoluent vite, donc l'équipe réalise des cycles de développement très rapides pour s'adapter au changement.
- La communication avec le client est essentielle, donc le client fait partie de l'équipe de développement.

- La communication au sein de l'équipe est primordiale, donc l'équipe est réunie dans un même bureau.

De plus, XP propose un pragmatisme extrême. Les pratiques qui apportent de la valeur au client doivent être appliquées. Le reste doit être amélioré et prouver son efficacité ou être supprimé en tant que gaspillage.

Mais où est l'originalité?

Les pratiques d'XP sont connues et appliquées depuis longtemps par certains développeurs. L'originalité réside dans le fait de toutes les combiner et de pousser leur application à l'extrême.
Aussi, la méthode accorde une très grande importance aux aspects humains du développement logiciel.
Enfin, la méthode est ancrée sur une vraie philosophie de développement décrite en termes de valeurs, principes et pratiques.

Quelles sont les valeurs?

XP affirme que l'efficacité maximale peut être atteinte si le développement respecte cinq valeurs: communication, retour d'information, simplicité, courage et respect.
La **communication** est ce qui importe le plus au sein d'une équipe de développement. Les problèmes rencontrés lors d'un projet sont souvent dus à des manques d'échange. Alors, la méthode préconise une communication intense et en continue au sein de l'équipe y compris avec le client.
La **simplicité** permet d'éviter les gaspillages et de rendre le logiciel souple, tolérant aux modifications. Ainsi, XP préconise de toujours choisir la solution la plus simple qui soit viable.
Le changement est inévitable et continu lors d'un développement logiciel. Plutôt que de gaspiller de l'énergie à contenir le changement, XP préconise de s'y adapter. Un **retour d'information** rapide et en continue sur le développement permet alors de piloter le travail de manière réactive et de l'adapter au changement.
Il faut du **courage** pour surmonter la peur de développer, changer sa manière de travailler, communiquer de manière transparente et confronter ses points de vues au retour d'information.
Enfin, le courage de l'équipe de développement mérite le **respect** de tous les intervenants du projet. De plus, un travail d'équipe efficace nécessite le respect mutuel des membres.

Les valeurs sont universelles et donc difficiles à traduire en pratiques. Par contre, les principes sont spécifiques au domaine. Ils aident à déterminer les pratiques concrètes à appliquer en regard des valeurs.

Quels sont les principes?

Humanité *(Humanity)*[1]
Les logiciels sont développés par des personnes. Il faut en tenir compte dans l'intérêt de l'équipe et de l'entreprise.
Economie *(Economics)*
Tout travail doit apporter de la valeur au client et au fournisseur.
Bénéfice partagé *(Mutual benefit)*
Toute activité doit bénéficier à toute personne impliquée.
Similitude *(Self-similarity)*
Une solution efficace peut être adaptée à différents contextes, mais sans garantie de succès.
Amélioration *(Improvement)*

1 *Les noms originaux en anglais des principes sont cités car ils constituent le vocabulaire de la méthode.*

Il faut constamment rechercher à s'améliorer.
Diversité *(Diversity)*
Il n'existe pas une unique et meilleure manière de procéder. C'est pourquoi l'équipe doit regrouper toutes les différentes compétences permettant le succès du projet et les membres doivent confronter leurs points de vues.
Réflexion *(Reflection)*
Une équipe efficace réfléchit à sa manière de travailler et apprend de ses inévitables erreurs.
Flux *(Flow)*
Il faut livrer un logiciel opérationnel en flux continu et non en phases séquentielles.
Opportunité *(Opportunity)*
Les problèmes doivent être considérés comme des opportunités d'apprendre et de s'améliorer.
Redondance *(Redundancy)*
Les problèmes critiques et difficiles doivent être résolus simultanément de différentes manières afin d'assurer une qualité maximale.
Echec *(Failure)*
Il vaut mieux échouer, apprendre et s'améliorer que ne pas avancer.
Qualité *(Quality)*
On ne pilote pas un développement en jouant sur la qualité du logiciel. Les projets n'avancent pas plus rapidement en abaissant le niveau de qualité. Aussi, les projets n'avancent pas plus lentement en imposant un niveau de qualité exigeant.
Petits pas *(Baby steps)*
Il faut développer par petits pas par simplicité et pour obtenir rapidement et continuellement des retours d'information.
Responsabilité acceptée *(Accepted Responsability)*
On ne peut imposer la responsabilité. Elle doit être acceptée.

Pour respecter les cinq valeurs, XP propose d'appliquer un ensemble cohérent de pratiques guidées par les principes.

Quelles sont les pratiques?

Il s'agit de la manière de travailler au jour le jour. Chaque pratique est applicable telle quelle. Néanmoins, elles peuvent aussi être considérées comme idéales et donc être adaptées à l'environnement concret du projet.

XP propose en premier un jeu de pratiques principales qui permettent de se mettre à XP.[2]
Travail en binôme *(Pair programming)*
Tout code est écrit à deux développeurs devant un même poste de travail afin de revoir le code en continue et de privilégier la communication. Les paires permutent. Cette organisation favorise la communication, le retour d'information et le respect.
Espace commun de travail *(Sit together)*
L'équipe partage un même bureau afin de privilégier la communication orale face à face et le travail en binôme.
Equipe complète *(Whole team)*
Afin d'accélérer la communication, l'équipe regroupe toutes les personnes dont les compétences permettent le succès du projet.
Management visuel *(Informative workspace)*
L'avancement et les difficultés du projet sont affichés dans l'espace de travail afin de faciliter la communication et le retour d'information.
Travail dynamique *(Energized work)*

2 *Les noms originaux en anglais des pratiques sont cités car ils constituent le vocabulaire de la méthode.*

Le rythme de travail doit être durable à long terme. De plus, la discipline de travail doit se plier aux contraintes du bureau partagé, du travail d'équipe et en binôme tout en respectant un espace personnel.

Scenarios *(Stories)*
La planification du travail est basée sur la quantification de fonctionnalités. Les fonctionnalités sont décrites en scénarios d'utilisation pertinents pour le client. L'objectif est de faciliter la communication entre client et développeurs.

Cycle hebdomadaire *(Weekly cycle)*
Le travail est planifié à la semaine. Un incrément de fonctionnalité est produit et livré en une itération d'une semaine. Ce cycle court favorise le retour d'information rapide et continue.

Cycle trimestriel *(Quarterly cycle)*
Le travail est planifié au trimestre. Ce cycle long permet de prendre du recul, de réfléchir à l'amélioration continue des pratiques et de traiter les goulots d'étranglement du flux de travail.

Du mou *(Slack)*
Il vaut mieux réduire le périmètre des engagements plutôt que de ne pas les tenir. Ceci permet de tenir un rythme durable et de maintenir un climat de confiance et de respect.

Construction en dix minutes *(Ten-minute build)*
Le système entier doit être automatiquement construit et testé en dix minutes. Cette pratique assure un retour rapide d'information.

Intégation continue *(Continuous integration)*
L'équipe entretient tôt et toujours une version testée et opérationnelle du système qui contient les toutes dernières modifications. Cette pratique assure également un retour rapide d'information.

Tester d'abord *(Test-first programming)*
Il faut écrire un test qui échoue avant toute modification de code. Ensuite, il faut changer le code pour faire passer le test. Ceci fournit un retour d'information très rapide sur la pertinence du code. Cela permet également d'écrire un code simple.

Conception incrémentale *(Incremental design)*
Il faut concevoir progressivement et continuellement améliorer et simplifier la conception. Ceci permet d'obtenir au plus vite un retour d'information sur les choix et de maintenir une conception simple et donc évolutive.

Ensuite, la méthode propose d'appliquer les pratiques corollaires une fois que les pratiques principales ont contribué à améliorer l'efficacité du développement.

Réelle implication du client *(Real customer involvement)*
Afin de favoriser la communication, le retour d'information et le respect mutuel, le client doit s'impliquer dans le développement, notamment en faisant partie de l'équipe, en partageant son espace de travail, en menant la planification, en fournissant des scénarios d'utilisation et des jeux de tests de recette.

Déploiement incrémental *(Incremental deployment)*
Afin d'obtenir un retour d'information et d'apporter au plus tôt de la valeur au client, il faut déployer en production le système par incréments tout au long du projet.

Equipe stable *(Team continuity)*
Afin d'assurer la communication, il faut conserver les équipes efficaces.

Réduire progressivement les équipes *(Shrinking teams)*
Au fur et à mesure que l'équipe s'améliore et gagne en efficacité, on peut réduire l'effectif pour former d'autres équipes. Ceci permet de communiquer la méthode au sein d'une organisation.

Analyse cause-racine *(Root-cause analysis)*
Lorsqu'un défaut est détecté, il faut corriger la cause du problème afin que ce type d'erreur ne soit plus commis.

Code partagé *(Shared code)*
N'importe qui dans l'équipe peut améliorer n'importe quel code du projet. L'amélioration continue du code est toujours une priorité. Cette pratique permet d'entretenir une implémentation simple et de communiquer à tous l'intention de tout le code.

Code et tests *(Code and tests)*
Afin d'organiser le travail de la manière la plus simple possible, il ne faut maintenir que le code et les tests. Tous les autres produits du développement doivent être générés à partir du code et des tests.
Base de code unique *(Single code base)*
Afin de favoriser la simplicité, il faut à tout prix éviter de gérer plusieurs versions du logiciel.
Déploiement quotidien *(Daily deployment)*
Pour obtenir des retours d'information, il faut déployer en production le logiciel développé dans la journée.
Contrat à contour négocié *(Negociated scope contract)*
Il faut tenir les objectifs de délai, de coût et de qualité en négociant le contour fonctionnel du logiciel. Les contrats doivent être adaptés à cette démarche.
Paiement à l'utilisation *(Pay-per-use)*
Il faut privilégier un paiement du fournisseur à l'utilisation du logiciel plutôt qu'un paiement aux livraisons. Ceci permet d'organiser le développement de manière à maximiser la valeur apportée au client.

Quel est le cycle de développement?

Le cycle de développement est itératif et incrémental. Une itération débute par une réunion de planification à laquelle participe le client et les développeurs. Le client décide des fonctionnalités prioritaires. Elles sont décrites sous la forme de scénarios d'utilisation. Les développeurs découpent les scénarios en tâches qu'ils quantifient. Avec cette mesure, l'équipe négocie l'incrément de fonctionnalité qu'elle s'engage à produire dans l'itération. Les tests automatisés d'acceptation sont écrits. Les binômes se forment et choisissent des tâches à implémenter. Lorsque les tests d'acception passent, le logiciel enrichi du nouvel incrément est livré.

En quoi XP aide à développer le bon logiciel?

Le client est impliqué tôt et régulièrement dans le développement. Les itérations fréquentes lui fournissent des versions fonctionnelles qui lui permettent d'affiner son besoin et de communiquer sa satisfaction. En quelque sorte, le développement est asservi sur le besoin et satisfaction du client.

En quoi XP aide à bien développer le logiciel?

Les relectures continues en binôme, les tests systématiques, le remaniement et le partage du code assurent qu'un haut niveau de qualité est maintenu.

En quoi XP aide à développer vite?

L'équipe ne réalise que des activités qui apportent de la valeur au client. L'intégration continue instaure un rythme soutenu de développement en équipe et permet de corriger tôt les dysfonctionnements liés au travail en parallèle. Enfin, le gaspillage est aussi et surtout évité en développant bien le bon logiciel.

Et l'homme dans tout cela?

XP a l'originalité d'accorder une importance primordiale à l'homme dans le développement logiciel. En effet, la méthode

- accorde à l'équipe la fierté du travail accompli et bien fait en instaurant des livraisons fonctionnelles fréquentes et en exigeant un haut niveau de qualité;
- stimule la solidarité et l'esprit d'équipe en regroupant les développeurs et systématisant

le travail en binôme;
- assure la formation continue et le partage des compétences dans le cadre du travail en binôme et les équipes pluridisciplinaires;
- supprime la peur de développer et accorde le droit à l'erreur en systématisant les petits pas, le test, l'intégration continue et le retour rapide d'information;
- exige un rythme soutenable pour l'équipe en limitant la durée de travail et en recommandant la réduction négociée de contour fonctionnel plutôt que le sprint;
- instaure des rapports sains entre client et fournisseur en créant une équipe intégrée avec communication transparente;
- accorde à l'équipe la liberté de s'organiser elle-même et d'améliorer ses propres pratiques;
- respecte le travail en supprimant le gaspillage par un cycle de développement en flux tendu.

Comment se mettre à XP?

La progression pour transiter vers XP dépend de l'appétit de changement de l'équipe. La méthode conseille de constater les bénéfices des pratiques principales avant de s'attaquer pleinement aux pratiques corollaires. On peut commencer par montrer l'exemple en appliquant soi-même certaines pratiques et en s'appuyant sur l'expérience de la communauté des praticiens[3]. Certaines équipes choisissent d'être guidées par un « coach ».

Faut-il tout pratiquer?

Il n'est pas nécessaire de tout appliquer à la lettre pour être un praticien. La méthode est un guide pour s'améliorer, gagner en efficacité et apporter de la valeur au client. Néanmoins, certains recommandent d'expérimenter une pratique extrême de XP pour ressentir l'alchimie qui née et pouvoir ensuite optimiser les pratiques à son environnement en connaissance de cause.[4]

XP est-il applicable pour tout type de développement?

La méthode est appliquée pour du développement Internet comme pour élaborer des logiciels temps-réel critiques embarqués pour l'avionique. Il s'agit d'adapter les pratiques à son contexte. Cette démarche est facilitée par la nature adaptative d'XP.

XP se limite t-il au développement de logiciels?

Même si les valeurs de XP sont universelles, les principes et les pratiques sont spécifiques au développement logiciel. Par contre, XP a des répercussions en dehors de ce cadre. En effet, l'amélioration continue de l'efficacité peut déplacer le goulot d'étranglement du flux de travail hors du périmètre du développement logiciel, en amont ou en aval. Pour étendre XP, on peut alors s'inspirer de méthodes proches, comme le Lean[5].

XP est-il lié à la démarche orientée-objet?

Un développement XP n'est pas nécessairement orienté-objet. Néanmoins, la méthode se base sur la construction incrémentale du logiciel, la tolérance aux changements et la testabilité. Or, ces objectifs sont plus aisément atteints par une pratique de l'encapsulation, de l'héritage et du polymorphisme.

---

3 *Comme la communauté http://fr.groups.yahoo.com/group/xp-france/*

4 *Voir: http://martinfowler.com/articles/xpVariation.html*

5 *Voir http://www.threeriversinstitute.org/LearningFromLean.html*

Est-ce compatible avec le développement offshore? Est-ce applicable pour de grandes équipes?

Toute méthode de travail d'équipe souffre de la perte de communication et réactivité qui accompagne les grandes équipes et les développements multisites[6]. Néanmoins, comme XP met l'accent sur la communication, la réactivité et la synchronisation fréquente, elle demeure efficace dans ces cadres. Les valeurs et principes restent pertinents. Il s'agit alors d'adapter les pratiques.[7]

XP est-il un retour en arrière?

Tout comme certaines des premières démarches de développement logiciel, XP est une méthode adaptative et empirique, qui parallélise les activités. Cela s'oppose aux méthodes qui ont suivi et qui se veulent théoriques et prédictives, et qui sérialisent les activités, comme le cycle en V. Ce n'est pas un retour au développement chaotique, mais plutôt un développement qui s'adapte au chaos intrinsèque à toute démarche de développement logiciel.

Perd t-on en visibilité et en prédictibilité?

L'estimation, la prévision et la planification théorique et massive en début de projet n'offrent qu'une illusion de visibilité et de prédictibilité. En mode XP, ces activités sont menées progressivement au fil des itérations à partir de retours d'information de plus en plus pertinents. Ainsi, une vraie visibilité et prédictibilité s'affinent avec le temps de manière empirique et fiable. Mais, il y a plus important: la méthode assure que le développement est asservi sur un parcours qui minimise le gaspillage et donc réduit la durée du projet.

La conception a t-elle disparue?

La conception massive en phase amont de l'implémentation est remplacée par une conception progressive et continuellement améliorée au fil des itérations. L'idée est de ne pas prendre d'hypothèse en avance de phase mais de concevoir en flux tendu le minimum suffisant pour les besoins actuels. L'évolutivité de la conception est assurée par sa simplicité et l'échafaudage de tests construits autour du code.

XP est-il compatible des démarches de qualité certifiantes ?

XP privilégie la communication informelle et l'investissement dans les tests et le code à une production de preuves documentaires, recherchées par les démarches de qualité certifiantes telles que ISO, CMMi, la DO178B ...
Ainsi, si une certification devient un objectif qui apporte de la valeur, il convient d'adapter la démarche XP afin de produire les preuves auditées. Bien sûr cela représente un supplément d'activité.

XP est-il sectaire?

Parce que les équipes XP sont soudées et changent radicalement leur manière de travailler et certainement aussi à cause du nom de la méthode, certains non-praticiens se sentent exclus et caricaturent ceux qui l'appliquent. D'ailleurs, il existe un débat intéressant sur ce sujet: XP partagerait certains attributs avec les sectes ou la méthode serait plutôt un mouvement en

6 *Voir The Mythical Man-Month: Essays on Software Engineering de Frederick P. Brooks*
7 *Voir http://martinfowler.com/articles/agileOffshore.html*

croissance.[8]

Comment les étudiants appréhendent-ils XP?

XP attire les étudiants. Le nom, l'auto-organisation, l'esprit d'équipe et la volonté de marquer une rupture avec les processus conventionnels séduisent l'esprit rebelle des jeunes. De plus, l'intégration de nouveaux embauchés par le travail en binôme et les pratiques visant à supprimer la peur de l'erreur (comme les tests et l'intégration continue) rassurent les futurs professionnels. Ils sont conscients qu'avec une telle philosophie de travail ils seront tout de suite parrainés et impliqués dans toutes les activités du développement plutôt que d'être cantonnés à une tâche de débutant. Cette attirance est positive car cela signifie que des développeurs débutent leur vie active avec l'envie d'expérimenter en équipe un développement logiciel efficace et humain dans un cadre qui privilégie l'amélioration continue.

Comment se renseigner davantage?

- Livre en anglais sur la 2nde édition: Kent Beck, Extreme Programming Explained (ISBN 0-321-27865-8).
- Livre en français sur la 1ère édition: Jean-Louis Bénard, Laurent Bossavit, Régis Médina et Dominic Williams, Gestion de projet eXtreme Programming (ISBN 2-212-11561-X)
- Groupe de discussion en français: http://fr.groups.yahoo.com/group/xp-france/

Emmanuel CHENU
emmanuel.chenu@fr.thalesgroup.com
http://emmanuelchenu.blogspot.com/

8 *Voir* http://www.hacknot.info/hacknot/action/showEntry?eid=11 *et* http://klimek.box4.net/blog/2007/02/26/xp-cult-or-movement/