Un point sur l'Extreme programming

Ou méthodes agiles

Nicolas Trèves

J'ai participé aux journées XP Day France qui se sont tenues à Paris les 23 et 24 mars 2006.
Ces journées ont été organisées sous la tutelle de l'association XP France qui a pour vocation de promouvoir en France l'utilisation des méthodes agiles, en particulier l'Extreme Programming. Cette association est affiliée à l'association internationale « agile alliance ». À ce titre, elle a organisé pour la première fois en France la conférence XP Day dans sa version française après que d'autres journées eurent lieu à Londres, Bruxelles, Rotterdam et Karslruhe. XP Day s'adresse aux professionnels du logiciel, quel que soit leur niveau de connaissance de l'Extreme Programming.
Avant de vous faire part de mes impressions, peut-être est-il utile de replacer ces journées dans leur contexte, certains d'entre nous n'ayant pas connaissance du domaine.

Ce que sont les méthodes agiles, selon Wikipedia

Une **méthode agile** est une méthode de développement informatique permettant de concevoir des logiciels en impliquant au maximum le demandeur (client), ce qui permet une grande réactivité à ses demandes. Les méthodes agiles se veulent plus pragmatiques que les méthodes traditionnelles. Elles visent la satisfaction réelle du besoin du client, et non d'un contrat établi préalablement. La notion de méthode agile est née à travers un manifeste signé par 17 personnalités, créateurs de méthodes ou dirigeants de sociétés.
Dans ce but, les méthodes Agiles prônent 4 valeurs fondamentales (entre parenthèse, les citations du manifeste) :

- L'équipe (« Personnes et interaction plutôt que processus et outils ») : Dans l'optique agile, elle est bien plus importante que les moyens matériels ou les procédures. Il est préférable d'avoir une équipe soudée et qui communique composée de développeurs moyens plutôt qu'une équipe composée d'individualistes, même brillants. La communication est une notion fondamentale.
- L'application (« Logiciel fonctionnel plutôt que documentation complète ») : Il est vital que l'application fonctionne. Le reste, et notamment la documentation technique, est secondaire, même si une documentation succincte et précise est utile comme moyen de communication. La documentation représente une charge de travail importante, mais peut pourtant être néfaste si elle n'est pas à jour. Il est préférable de commenter abondamment le code lui-même, et surtout de transférer les compétences au sein de l'équipe (on en revient à l'importance de la communication).
- La collaboration (« Collaboration avec le client plutôt que négociation de contrat ») : Le client doit être impliqué dans le développement. On ne peut se contenter de négocier un contrat au début du projet, puis de négliger les demandes du client. Le client doit collaborer avec l'équipe et fournir un feed-back continu sur l'adaptation du logiciel à ses attentes.
- L'acceptation du changement (« Réagir au changement plutôt que suivre un plan ») : La planification initiale et la structure du logiciel doivent être flexibles afin de permettre l'évolution de la demande du client tout au long du projet. Les premières releases du logiciel vont souvent provoquer des demandes d'évolution.

Ces 4 valeurs se déclinent en 12 principes généraux communs à toutes les méthodes agiles :

- « Notre première priorité est de satisfaire le client en livrant tôt et régulièrement des logiciels utiles ».
- « Le changement est bienvenu, même tardivement dans le développement. Les processus agiles exploitent le changement comme avantage compétitif pour le client ».
- « Livrer fréquemment une application fonctionnelle, toutes les deux semaines à deux mois, avec une tendance pour la période la plus courte ».
- « Les gens de l'art et les développeurs doivent collaborer quotidiennement au projet ».
- « Bâtissez le projet autour de personnes motivées. Donnez-leur l'environnement et le soutien dont elles ont besoin, et croyez en leur capacité à faire le travail ».
- « La méthode la plus efficace de transmettre l'information est une conversation en face à face ».
- « Un logiciel fonctionnel est la meilleure unité de mesure de la progression du projet ».
- « Les processus agiles promeuvent un rythme de développement soutenable. Commanditaires, développeurs et utilisateurs devraient pouvoir maintenir le rythme indéfiniment ».
- « Une attention continue à l'excellence technique et à la qualité de la conception améliore l'agilité ».
- « La simplicité - l'art de maximiser la quantité de travail à ne pas faire - est essentielle ».
- « Les meilleures architectures, spécifications et conceptions sont issues d'équipes qui s'auto-organisent ».

- « À intervalle régulier, l'équipe réfléchit aux moyens de devenir plus efficace, puis accorde et ajuste son comportement dans ce sens ».

Les méthodes Agiles figurent au nombre de 7 :
- Adaptive software development (ASD)[1] ;
- Crystal clear[2];
- Dynamic software development method (DSDM)[3];
- Extreme programming (XP)[4];
- Feature driven development[5];
- Rapid Application Development (RAD)[6] ;
- Scrum[7].

Extreme programming

L'**Extreme Programming** (XP) est une méthode agile de gestion de projet informatique, dont l'objectif est de permettre de gérer des projets de manière simple et efficace. L'Extreme Programming est née officiellement en octobre 2000 (voir référence Beck en fin d'article).

Comme toutes les méthodes de développement, l'Extreme Programming propose un cadre pour l'ensemble des aspects du projet logiciel, depuis l'analyse des besoins jusqu'aux tests, en passant par la conception. Mais à la différence des processus prédictifs, recourant généralement à UML (même si XP utilise aussi parfois UML comme support de communication), XP ne se fonde pas sur la définition exhaustive et précoce des besoins ; elle parie plutôt, à partir d'un ensemble de règles strictes, sur la souplesse et la mise en valeur du « capital humain ».

Tout en mettant l'accent sur les bonnes pratiques de programmation, XP préconise un déroulement par itération courte et géré collectivement, avec une implication constante du client. Il en découle une redéfinition de la relation entre client et fournisseur, avec de surprenants résultats en termes de qualité de code, de délais et de satisfaction de la demande du client.

L'Extreme Programming repose sur 4 valeurs fondamentales :
- **La 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. Si un manque apparaît malgré tout, un coach se charge de l'identifier et de remettre ces personnes en contact.
- **La simplicité**
  La façon la plus simple d'arriver au résultat est la meilleure. Anticiper les extensions futures est une perte de temps. Une application simple sera plus facile à faire évoluer.
- **Le feedback**
  Le retour d'information est primordial pour le programmeur et le client. Les tests unitaires indiquent si le code fonctionne. Les tests fonctionnels donnent l'avancement du projet. Les livraisons fréquentes permettent de tester les fonctionnalités rapidement.
- **Le courage**
  Certains changements demandent beaucoup de courage. Il faut parfois changer l'architecture d'un projet, jeter du code pour en produire un meilleur ou essayer une nouvelle technique. Le courage permet de sortir d'une situation inadaptée. C'est difficile, mais la simplicité, le feedback et la communication rendent ces tâches accessibles.

Ces 4 valeurs se déclinent en 13 pratiques :
- **Un représentant du client sur site**
  Afin d'assurer une meilleure réactivité et un retour rapide, un représentant du client doit, si possible, être présent pendant toute la durée du projet. Ce représentant doit avoir les connaissances d'un utilisateur final, mais en même temps une vision globale du résultat à obtenir.
- **Jeu de planning**
  Le client crée des scénarios d'utilisation, en les priorisant. Ces scénarios seront ensuite implémentés par l'équipe de développement. Le projet est considéré comme achevé quand le client n'est plus en mesure de trouver de nouveau scénario.
- **Intégration continue**
  L'intégralité de ce qui est développé est assemblée à chaque fois qu'une tâche est terminée, ce qui permet d'avoir toujours une version à jour, notamment comme livrable ou pour le démarrage de nouvelles tâches.
- **Petites livraisons**
  Les livraisons doivent être les plus fréquentes possibles, afin d'obtenir un retour le plus rapidement possible, tout en offrant des fonctionnalités complètes. La fréquence de livraison est donc de quelques semaines.
- **Rythme soutenable**
  Afin d'éviter de surcharger l'équipe, elle ne fait pas d'heures supplémentaires deux semaines de suite. Si le cas se présentait, cela signifierait qu'il faut redéfinir l'emploi du temps.
- **Tests de recette**
  À partir de critères définis par le client, on crée des procédures de test, souvent automatisées, qui permettent de vérifier fréquemment le bon fonctionnement de l'application.
- **Tests unitaires**
  Avant d'implémenter une fonctionnalité, il convient d'écrire un test qui validera l'implémentation. Ces tests sont issus de la traduction à plus bas niveau des Tests de recette. Il est donc normal qu'XP pousse la notion de Tests unitaires (Unit Tests) en

[1] *www.adaptivesd.com*
[2] *www.informit.com*
[3] *www.dsdm.org*
[4] *www.xprogramming.com*
[5] *www.featuredrivendevelopment.com*
[6] *www.rad.fr*
[7] *www.controlchaos.com*

imposant leur écriture avant celui du code (first-test).

- **Conception simple**
  L'objectif est de produire une application qui réponde aux attentes du client. Mieux vaut donc arriver à ce résultat de la manière la plus simple possible, afin de faciliter les évolutions futures en rendant le code simple à comprendre et facile à modifier. De même, la documentation doit être minimale, c'est-à-dire ce qui est demandé par le client.
- **Utilisation de métaphores**
  On utilise des métaphores et des analogies pour décrire le système et son fonctionnement, ce qui permet de le rendre compréhensible par les non-informaticiens, comme les utilisateurs ou les commerciaux.
- **Remaniement du code** (ou refactoring)
  Amélioration continue de la qualité du code sans en modifier le comportement (factorisation, commentaire du code, respect des règles de nommage, simplification, etc).
- **Convention de nommage**
  Dans l'optique d'appropriation collective du code et de facilitation de la communication, il est indispensable d'établir et de respecter des normes de nommage pour les variables, méthodes, objets, classes, fichiers, etc.
- **Programmation en binôme**
  La programmation se fait par deux. Le premier, appelé driver, a le clavier. C'est lui qui va travailler sur la portion de code à écrire. Le second, appelé partner, est là pour l'aider, en suggérant de nouvelles possibilités ou en décelant d'éventuels problèmes. Les développeurs changent fréquemment de partenaires, ce qui permet d'améliorer la connaissance collective de l'application et d'améliorer la communication au sein de l'équipe.
- **Appropriation collective du code**
  L'équipe est collectivement responsable de l'application et est supposée connaître l'intégralité du code. Chacun peut également faire des modifications dans toutes les portions du code, même celles qu'il n'a pas écrites.

D'autres principes régissent également l'Extreme Programming :

- ne pas ajouter de fonctionnalités plus tôt que prévu ;
- n'optimiser qu'à la toute fin.

Cette méthode s'appuie donc sur :

- une forte réactivité au changement des besoins du client ;
- un travail d'équipe ;
- la qualité du code ;
- la qualité des tests effectués au plus tôt.

Les journées XP France

Ces journées étaient organisées en sessions plénières et en sessions parallèles, autour de 9 thèmes principaux : pratiques techniques, management d'équipes, « Retours d'Expérience », « Excellence Technique », « Stratégies de Projet », « Management de Produits », « Management des Personnes », « Horizons de l'Agilité », « Agile à grande échelle ».

J'ai suivi les sessions qui portaient surtout autour des retours d'expérience et du management de projets.

Ce que j'ai pu ressentir :

- 100 personnes environ étaient présentes.
- Le public qui assistait à la conférence est jeune, de moyenne d'âge environ de 40 ans. Il s'agit en majorité de programmeurs experts ou confirmés. Petites sociétés, type SSII.
- Ils appliquent les méthodes agiles dans leur contexte professionnel.
- Il y a une réelle pénétration de ces concepts dans la communauté des développeurs. Cependant pour que ça marche, il faut que les Directions Générales soient convaincues.
- L'accent était mis sur le développement conduit par les tests et le développement par binôme.
- Du point de vue de la gestion de projet, le développement par binôme est coûteux mais chiffres à l'appui, il est démontré qu'il est totalement efficace si l'ensemble des principes sont suivis dans un projet.
- Des discussions ont également porté sur le niveau de maturité des organisations requis pour maîtriser l'Extreme Programming. L'utilisation des processus du CMMI a été évoquée.
- L'agilité pour tous les projets, en particulier les développement à grande échelle, type ingénierie/intégration de systèmes est-elle possible ? L'agilité partout n'est peut-être pas systématique mais elle peut s'appliquer à des sous-ensemble de projets.
- Rien n'a porté sur la conception/modélisation.
- Il a été conclu que l'Extreme Programming est particulièrement adapté à la réalisation d'applications s'appuyant sur des architectures standard de type open source ou environnment java (J2EE…), .NET, etc, mais également a montré des promesses dans des projets de type EAI[1].

[1] *EAI :* « Enterprise Application Integration »*, en français :* intégration des applications de l'entreprise

Ces journées se sont closes sur le constat qu'il s'agit d'une méthode qui ne fait pas l'unanimité.

- L'Extreme Programming est parfois critiqué pour sa radicalité. En effet, si on en croit les chantres de cette méthode, et notamment ses auteurs, on ne fait de l'Extreme Programming que si l'on met en place **toutes** les pratiques de la méthode, ce que certains trouvent trop contraignant et contraire à l'esprit agile, qui doit justement permettre d'assouplir l'aspect méthodologique des projets informatiques.
- On reproche aussi parfois à cette méthode d'être mal adaptée à l'environnement économique : comment prévoir la durée et le coût d'un projet ?
- Enfin, on peut lui reprocher aussi de constituer essentiellement une refonte de méthodes largement utilisées depuis plus de trente ans. ▲

Nicolas.treves@adeli.org

Pour plus d'information, voir notamment :

- http://xp-france.net/
- http://fr.wikipedia.org/wiki/Méthode_agile
- Ecosystème des projets informatiques : agilité et discipline. Printz - Hermes. ISBN : 978-2-7462-1145-2 (12/2005)
- Gestion de projet Extreme Programming. Benard - eyrolles. Isbn : 978-2-212-11561-1 (11/2004)
- Extreme Programming explained: embrace change 2nd edition. Beck - Addison Wesley. ISBN : 978-0-321-27865-4 (12/2004)