À propos d'agilité

Interview de Jean-Pierre Vickoff

Rapporté par Martine Otter

Jean-Pierre Vickoff a mis en œuvre les techniques dites « agiles » dès 1991 au Canada. Il nous a fait la démonstration de leur efficacité, y compris en matière de forfait, lors d'une rencontre ADELI le 18 mars 2013.

Jean-Pierre Vickoff, qui êtes-vous ?

Jean-Pierre Vickoff indique qu'il pratique « l'agilité » telle que définie actuellement depuis 1991, date à partir de laquelle, alors au Québec, il s'est intéressé aux méthodes de développement incrémentales et itératives et les a appliquées au développement de logiciel.
Dès 1994, il fut ensuite le promoteur en France de la méthode RAD[1] (Rapid Application Development) dont découlent aujourd'hui les diverses méthodes agiles, par exemple SCRUM et XP. Jean-Pierre Vickoff proposa ensuite la méthode PUMA (Processus Urbanisant les Méthodes Agiles) dont les principales avancées ont été publiées dans la Lettre d'ADELI entre 2002 et 2013 (n°48, 68, 74 et 91) et sont disponibles sur le site d'ADELI (www.adeli.org).

Jean-Pierre, forfait et agilité, c'est contradictoire, non ?

Comment faire des projets agiles forfaitaires ?
C'est effectivement un véritable défi. Il ne suffit pas de faire des vœux pieux du style « informaticiens et clients doivent se faire confiance mutuellement ».

La réussite d'un projet forfaitaire, que ce soit en mode classique ou en mode agile, suppose la présence de trois éléments : des exigences, une estimation de la charge de réalisation et un reporting convenable. Ce dernier point implique une métrique formelle des changements imposés par le client (propriétaire du produit).

Comment définir les exigences ?

En mode classique, le client fournit théoriquement au départ un document « cahier des charges », à partir duquel une estimation est réalisée.
En mode agile, les exigences sont exprimées progressivement sous forme de récits utilisateurs.
Le récit utilisateur expose la vision du besoin et de la solution. Il ne s'agit pas d'un use case utilisateur, tel qu'on le définit dans UML, mais bien d'un véritable récit sous forme textuelle.
L'histoire est opérationnellement formalisée sur une fiche (post-it) qui explique ce que veut obtenir l'utilisateur et comment il considère que cela doit se passer. La fiche servira aussi ensuite à contrôler le déroulement de sa réalisation. On y inscrit au recto le n° de l'exigence, son intitulé et un texte descriptif.

[1] *www.rad.fr*

Au verso on indique dès le départ comment l'utilisateur testera la fonctionnalité avant de l'accepter.

En mode agile, on n'entame, en effet, le développement que si les cas de test sont définis préalablement au début de leur réalisation.
À la fin de la production, on portera ensuite sur cette fiche le n° de l'incrément livré (ou n° du sprint si l'on adopte la méthode SCRUM[1] et sa terminologie).

Mais il n'y a pas que des exigences fonctionnelles racontables sous forme d'histoire. Comment traite-t-on les autres exigences ?

Je vous propose une structuration agile des exigences, fruit de mon expérience du conseil, sous forme de 4 classes d'exigences :

- exigences des contraintes ;
- exigences fonctionnelles ;
- exigences techniques ;
- exigences organisationnelles.

Elles s'exprimeront sur 4 niveaux de profondeur selon l'importance du projet. Un seul impératif : les exigences doivent être rassemblées dans un document UNIQUE (il n'y aura bien qu'un seul code à la fin ?).

Et l'estimation de charge ?

La charge à produire du contrat est estimée en unités d'œuvre (UO). On parle de journées idéales : il ne s'agit pas de journées travaillées réelles, mais d'un temps de travail « espéré ».

Dans les méthodes classiques, la charge à produire est généralement pondérée en fonction de l'environnement technique et de facteurs liés aux ressources humaines.
Dans les méthodes agiles, on ne pondère pas formellement, on demande l'avis de ceux qui vont faire. C'est cet avis qui détermine le nombre de journées idéales, qui pourra être, au final, différent à la fin du réel consommé, les développeurs pouvant aller plus ou moins vite.

Jean-Pierre Vickoff nous rappelle au passage la différence entre un développement de type itératif et un développement incrémental.

L'incrémental peut très bien livrer des éléments qui, pour une raison ou une autre, deviendront inutiles, alors qu'une méthode vraiment itérative permet le retour arrière[2] et l'abandon ou la modification (avec réutilisation éventuelles) de parties déjà acceptées.

En pratique l'estimation des charges se fait sous forme d'un planning poker.

Chacun choisit une carte et la dépose à l'envers devant lui. Après avoir retourné les cartes, on confronte, en les argumentant techniquement, les différentes estimations.
On peut réitérer l'opération et, s'il n'y a toujours pas de consensus, on retient au final la moyenne des évaluations exprimées par les développeurs (sauf si des problèmes non traitables obligent à reporter la décision).

Notons que la charge à estimer doit être complète et comprendre de manière exhaustive toute la problématique d'un projet (incluant les problèmes administratifs, les absences et diverses pertes de temps).
La documentation basique en fait partie ainsi que toutes autres formes de livrables, si le besoin en a été exprimé formellement.
La correction des anomalies pourra de même nécessiter un dernier sprint (ou un site pilote), mais dans ce cas, il faudra le prévoir formellement.

[1] *SCRUM est la principale méthode agile actuelle.*

[2] *Pour des explications détaillées sur ces concepts, on pourra se reporter à l'article paru dans la Lettre n°91.*

En pratique, sur un forfait, ce ne sont pas les développeurs qui font les estimations, la charge leur est imposée. Est-ce bien réaliste ?

Il est exact que dans un contexte classique, on impose les charges au développeur qui ne s'en estime donc pas responsable. En mode agile, les développeurs s'engagent lors du planning poker. Ils sont donc responsables collectivement de l'estimation et de la réalisation qui en découle.
Le véritable point d'achoppement de l'estimation agile est alors la confiance dans l'honnêteté du planning poker auquel participeront le client et, éventuellement, un expert le représentant.
Les informaticiens peuvent se tromper ou souhaiter garder une marge de réserve….Il faut du respect de part et d'autre pour que la confiance soit réciproque.

Comment gère-t-on un projet agile ?

Comme sur tout projet, il y a des problèmes et des améliorations potentielles en cours de route.
On trace les problèmes dès le départ en les affichant sur le « reporting » mural au fur et à mesure de leur apparition : à chaque problème sont associés une date d'ouverture, une date de fermeture et un une personne « en charge de sa résolution ».
De même les améliorations à mettre en œuvre, détectées lors des rétrospectives, sont affichées ainsi que les points déterminants du projet.

L'avancement de la réalisation est représenté sous forme d'un Kanban. Le Kanban permet de visualiser les tâches selon leur statut : à faire (ce que l'on veut fabriquer), en cours, tâches abandonnées, livrées, testées (techniquement, fonctionnellement). Le Kanban est mis à jour en temps réel sur le tableau. Il est possible de visualiser immédiatement l'avancement du projet sans avoir à poser de question.

Cette approche permet, entre autre, de justifier à tout moment le temps passé et de faire le bilan des exigences livrées utiles ou des exigences éventuellement abandonnées par l'utilisateur.

La bonne entente ne suffit pas : ce qui est agile doit être également mesuré ; la confiance doit être justifiée.

Les conditions de la réussite

Les équipes du client et du fournisseur doivent rester proches les unes des autres, idéalement sur un même plateau. Il est illusoire d'envisager une réalisation agile en offshore, ni même de façon distante, tant que la technologie ne permettra pas de partager un mur complet permettant de visualiser des dizaines ou des centaines de post-it.

Au préalable, les règles du jeu d'un développement agile doivent être clairement exposées au client qui doit les comprendre et les accepter, faute de quoi ce dernier rechignera inévitablement à payer les exigences abandonnées.
La formation du client à l'agilité est indispensable. Si le client n'adhère pas aux principes de l'agilité et à l'esprit collaboratif qui le sous-tend, il est impératif de revenir à une méthode de développement classique.

Ce mode de fonctionnement ne relève-t-il pas de l'utopie ?

Nous ne sommes pas dans un monde idéal.
La plupart des sociétés de service exercent sur les équipes agiles une pression identique à celle qu'elles exercent sur les équipes classiques.

Le client doit rédiger son appel d'offres en spécifiant clairement les exigences d'un projet agile. Cela impliquera l'affectation de ressources à temps plein et la proximité entre équipes métiers et équipe de développement, réunies dans un même lieu. La charte agile doit être signée par le client et le fournisseur.

Un gain en qualité énorme peut être obtenu en mettant en place le « pair programming » (programmation par équipe de deux), préconisé par XP, mais dont le coût est de l'ordre de 30 % en réalisation pour des gains multiples ensuite en exploitation et maintenance.

Il est donc impératif que le client qui souhaite un code de qualité indique dans son appel d'offres que les développements doivent être réalisés en pair programming, sinon c'est le moins-disant qui l'emportera, au risque d'une qualité moindre et d'un coût global de possession plus élevé sur le long terme.

L'agilité ignore-t-elle les processus ?

Non, pour démarrer un projet de développement, il faut avoir une idée de ce que l'on veut obtenir et disposer d'une liste minimale de fonctionnalités à produire.
On prévoit généralement une première itération, le « sprint 0 » (SCRUM), pour établir cette liste : la modélisation des processus et éventuellement leur reengineering peuvent faire partie de cette étape préalable.
On peut passer au développement proprement dit dès que l'on dispose de suffisamment d'éléments pour assurer la charge à produire d'un incrément livrable (vélocité théorique initiale de l'équipe).

Tout n'est pas nécessaire pour commencer, mais disposer d'une vision globale incluant les interdépendances fonctionnelles et techniques est impératif.

Et les tests ?
Pratique-t-on des tests de non-régression ?

SCRUM ne s'intéresse pas au génie logiciel.
Il ne dit donc rien sur ce sujet. XP s'attache ? par contre, aux techniques les plus avancées de test et propose de coupler le principe du TDD (Test driven development, Test fail first) au pair programming et à la rotation des binômes ainsi qu'à l'intégration continue.
L'ensemble permet une maîtrise totale de chaque changement.

L'agilité ne favorise-t-elle pas la remise en cause permanente des exigences ?

Cela dépend des méthodes. Dans SCRUM chaque sprint se déroule sans changement du but initial.

L'agilité élargie (XP, PUMA) permet, par contre, de prendre en compte les nouvelles exigences qui émergent en cours de projet mais elle en tient une comptabilité, estimée en temps réel et acceptée formellement par le demandeur.

Le forfait agile est donc un engagement sur une vélocité acceptée par les informaticiens dans le cadre d'un périmètre fixe en terme d'unité d'œuvre mais dont le contenu fonctionnel est modifiable par l'utilisateur dans le cadre d'une métrique formelle de ce changement.

Conclusion

ADELI remercie Jean-Pierre Vickoff pour sa démonstration et retiendra qu'un projet agile peut être conduit au forfait dans un climat de confiance lorsqu'un dispositif de mesure permet de justifier cette confiance.▲

vickoff@noos.fr