Comment parler de l'agilité à des industriels?

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

En tant que praticiens du développement logiciel Agile dans un secteur industriel, nous avons souvent dû justifier, la pertinence de notre démarche de travail auprès de notre management.
Au cours de présentations et démonstrations, nous nous sommes rendu compte que notre hiérarchie était peu réceptive aux termes "eXtreme-Programming", "Scrum", "ScrumMaster" ou même "Agile". Nous avons même constaté que ces noms desservent parfois les démarches qu'ils nomment.
Par contre, nous avons remarqué que nos managers sont bien plus réceptifs à l'évocation de la démarche Lean, plus courante dans la culture industrielle.
Ainsi, nous avons appris à adapter notre communication au vocabulaire du Lean pour accrocher nos décideurs.
Voici comment nous communiquons sur l'Agilité, sans l'évoquer directement.

Quels sont les grands principes du Lean?

Les cinq principes les plus marquants du **Lean** sont:

- **Value**
  Du le point de vue du client, spécifier ce qui lui apporte de la valeur.
- **Value Stream**
  Identifier toutes les étapes dans le flux de valeur. Éliminer chaque étape, action et pratique qui ne créé pas de valeur.
- **Flow**
  Faire en sorte que les étapes restantes créatrices de valeur se déroulent dans une séquence serrée afin que le produit suive un flux fluide vers le client.
- **Pull**
  Laisser le client tirer la valeur le long du flux à sa demande.
- **Perfection**
  Alors que ces étapes conduisent à une plus grande transparence, permettant aux managers et aux équipes d'éliminer les gaspillages, rechercher la perfection par l'amélioration continue.

Comment cela se transpose au développement de logiciels dans nos équipes?

En regard de ces cinq principes, nous avons mis en place des pratiques de développement logiciel issues du mouvement Agile[1], de la méthode Scrum[2] et de l'eXtreme-Programming[3].

Value

- Les spécifications sont élaborées par le client ou son représentant interne en utilisant son propre vocabulaire.
- Pour s'assurer que ses besoins sont bien communiqués aux développeurs, le client fournit ses critères d'acceptation, idéalement sous la forme de jeux de tests de recette.
- Le client donne une priorité à ses besoins. Cette classification pilote la chronologie des travaux pour optimiser le retour sur investissement.

Value stream

- L'équipe définit son flux de valeur. Sa description est formalisée dans le plan de développement. En

1 Voir le **Manifeste Agile**, en bibliographie.
2 Voir le livre **Agile Project Management With Scrum**, en bibliographie.
3 Voir le livre **eXtreme Programing Explained**, en bibliographie.

pratique, le flux est affiché dans le bureau des développeurs sous la forme d'un grand "task board", parcouru par les besoins identifiés sur des post-its.
- Le développement d'un besoin est terminé si toutes les activités à mener sont terminées *(conception, codage, tests unitaires, tests d'acceptation, traçabilité, couverture, documentation, revues ...)*.
- Lors de courtes réunions quotidiennes et des rétrospectives d'itération, l'équipe recherche à identifier et éliminer toute activité qui n'apporte pas de valeur.

Flow

- Le traitement d'un besoin doit être terminé avant de débuter le traitement d'un nouveau besoin.
- Le développement est itératif et incrémental afin de créer des séquences continues et fluides de création de valeur. Les itérations sont courtes afin de rythmer la création de valeur. Les versions du logiciel sont livrées au client.
- L'équipe se synchronise très régulièrement afin d'éviter les pertes de productivité dues aux désynchronisations. L'équipe synchronise les activités en cours lors de courtes réunions d'avancement quotidien et synchronise les développements par une pratique de l'intégration continue.
- Les activités répétitives et demandant peu de réflexion sont automatisées afin d'accélérer le flux de travail.
- Introduction de défauts dans le produit est traitée de manière préventive par la pratique du développement piloté par les tests (TDD)[4], l'intégration continue[5], la programmation par contrat[6], le pair-programming[7] et les revues de pairs.
- Les régressions sont détectées par des tests automatisés qui préviennent l'équipe par mail.
- Les équipes sont pluridisciplinaires et l'espace de travail est partagé afin de rendre la communication fluide.

Pull

- Le client donne une priorité à ses besoins et définit le contenu des itérations à partir des estimations de l'équipe de développement. Cela optimise le retour sur investissement car l'équipe implémente au plus tôt les besoins qui apportent le plus de valeur. Cette démarche dé-risque les projets et évite les gaspillages.

Perfection

- Lors de courtes réunions quotidiennes d'avancement et lors de régulières rétrospectives d'itération, l'équipe cherche à améliorer ses pratiques. L'effet de ses décisions peut être objectivement mesuré car la santé du logiciel est quantifiée en continue *(par une automatisation des tests, une intégration continue, une mesure des performances du logiciel, une mesure de la couverture des tests, une mesure de la qualité du code ...)* et l'efficacité de l'équipe est quantifiée en continue *(mesure de la vélocité - ou productivité)*.
- Par soucis de transparence, ces indicateurs sont publiés *(par affichage et sur site intranet)*.
- La recherche de l'amélioration continue se concrétise notamment par la recherche de l'automatisation, le remaniement du code et l'entretien de standards évolutifs.

Conclusion

Cette démarche progressive et continue a été initiée pour le développement de logiciels. L'enjeu est désormais d'entendre son périmètre à tout le flux de valeur.

Bibliographie, pour aller plus loin

- Manifesto for Agile Software Development - http://agilemanifesto.org/

4 Voir le livre **eXtreme Programing Explained**, en bibliographie.
5 Voir le livre **eXtreme Programing Explained**, en bibliographie.
6 Voir le livre **Conception et programmation orientées objet**, en bibliographie.
7 Voir le livre **eXtreme Programing Explained**, en bibliographie.

- Peter Middleton et James Sutton - Lean Software Strategies, Proven Techniques for Managers and Developpers - ISBN 1-56327-305-5
- Kent Beck and Cynthia Andres – Extreme Programming Explained, embrace change – ISBN 0-321-27865-8
- Mary et Tom Poppendieck – Lean Software Development, An Agile Toolkit – ISBN 0-321-15078-3
- Mary et Tom Poppendieck – Implementing Lean Software development, From Concept to Cash.
- Ken Schwaber – Agile Project Management With Scrum – ISBN 0-7356-1993-X
- Bertrand Meyer - Conception et programmation orientées objet – ISBN 0-136291-55-4