



 **Introduction à l'Agile**
(22/01/2012)



- > Qui suis-je ?
- > Quelques notions sur l'Agile
- > Retour d'expérience sur des projets
- > Questions / ROTI*

*Return On Time Invested

- > Martin Bahier (mbahier@octo.com, @MartinBahier)

- > Études
 - + M2 Info @Paris Diderot (2009)
 - + MS Management de Projets Technologiques @ESSEC/Telecom Paris (2010)

- > Aujourd'hui
 - + Consultant @Octo Technology (conseil en SI)
 - o Développement
 - o Audit



Quelques notions sur l'Agile

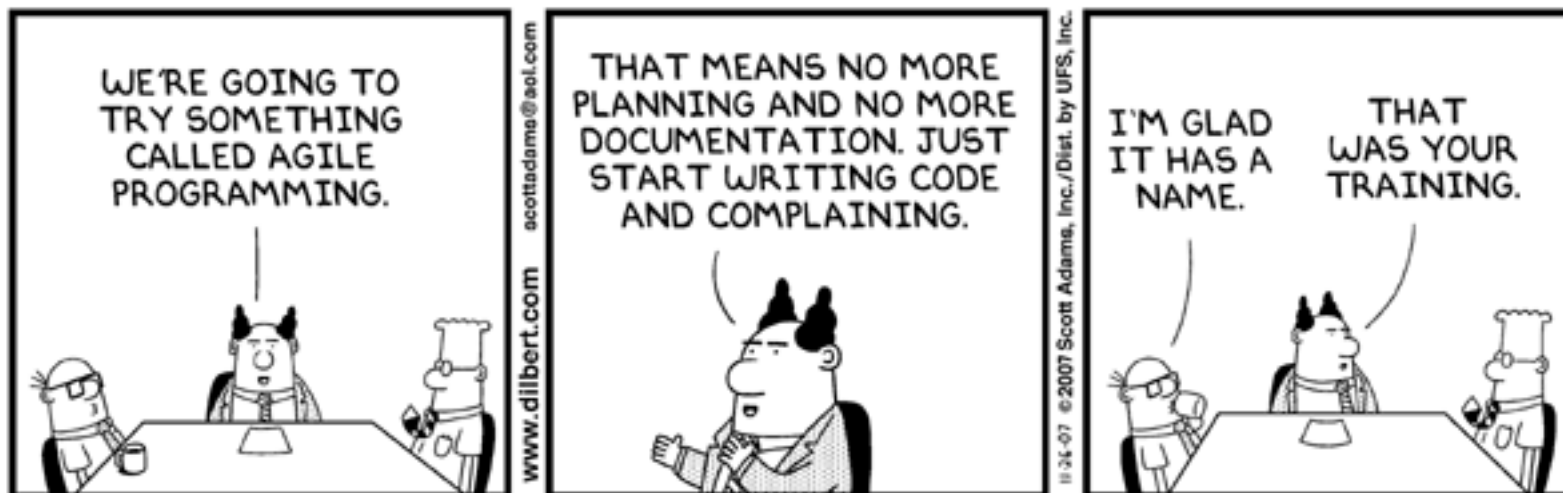
- > Le terme « Agile » apparaît vers 2000
- > Outils de l'industrie adaptés au développement logiciel
- > Inspiration du lean management et du Toyota Production System

Objectif : S'adapter au changement pour produire de meilleurs logiciels

- > Assemblage incrémental
- > Pilotage à la valeur métier : Prioriser le développement (fabriquer le plus utile le plus tôt)
- > Livrer un logiciel qui correspond aux attentes
- > Un logiciel bien conçu techniquement
- > Des démonstrations fréquentes (validation en continu, « fail fast »)
- > Des équipes de petite taille (10 personnes = beaucoup)

- > La méthode de gestion de projet Agile la plus répandue aujourd'hui
- > 4 Rituels
 - + Sprint planning (au début d'une itération)
 - + sprint review (en fin)
 - + Sprint reto
 - + Daily scrum
- > 3 Roles
 - + Product Owner : ordonne les commandes et priorise
 - + SCRUM master : valide le bon déroulement, aide l'équipe
 - + Équipe de dev : toutes personnes qui aident à la réalisation du produit
- > 2 Artefacts
 - + Produit
 - + Backlog (« carnet de commandes »)
- > 1 Définition : DoD (« comment je sais que j'ai fini ? »)
- > Scrum.org

- > « Adapter en permanence » ne veut pas dire « Tout accepter »
- > « Produire une application de qualité » ne veut pas dire « N'intégrer les changements que lorsqu'ils sont 'parfaits' »
- > « Rédiger les spécifications au fil de l'eau » ne veut pas dire « Improviser collectivement »



Travailler en Agile, c'est une responsabilité partagée ...

- > ... par l'équipe de développement
 - + Assurer la qualité, les livraisons / démonstrations, le périmètre
- > ... par le client du projet / le PO
 - + Respecter la charge de l'équipe de développement
 - + Entretenir le Backlog du projet
 - + Être aux rendez-vous communs
 - + Participer pro-activement au projet



Retours d'expérience



REX 1 : Une appli pour le monde de la santé

- > Domaine métier / tech nouveau pour nous
 - + La santé en France / en Europe
 - + Phases d'homologation ne dépendant pas que du client
 - + Prise en compte d'outils comme les lecteurs de cartes des médecins / carte Vitale
- > Le projet dure depuis plus de 2 ans et ça continue
- > L'équipe a du changer presque complètement entre chaque phase pour diverses raisons
- > Le profil des utilisateurs finaux (et leur nombre) a beaucoup évolué
- > Le périmètre fonctionnel a beaucoup évolué


- > PO virtuellement absent
 - > Client invasif (perturbe l'équipe de dev, souhaite tout changer à 2 jours de la livraison ...)
 - > L'équipe a réussi à livrer un logiciel correspondant aux attentes en concertation avec le client
 - > Toutes les briques fonctionnelles ne sont pas connues du client, le besoin et les moyens de le réaliser arrivent par petites touches
 - > Création d'une nouvelle application (virage fonctionnel)
-
- > Take away : Fonctionner incrémentalement mitige les risques dus au trop grand manque d'information.

Merci !

Questions ?

Merci !

ROTI



REX 2 : Un projet perso

- > Développement d'une application de streaming audio
- > Réalisation pendant le temps libre des développeurs
- > L'équipe assure toutes les fonctions à la fois (PO / dev / client ...)
- > Objectifs fonctionnels (on veut fabriquer qqch qu'on va utiliser) mais aussi tech (on veut se faire plaisir niveau code).
- > On souhaite diffuser le code une fois qu'on aura quelque chose de « sec » (v1) pour que des contributeurs puissent se greffer sur le projet.

- > Contributions complètement asynchrones mais on trouve le temps de discuter ensemble de la direction à faire prendre au projet
- > Plus difficile d'utiliser des outils visuels comme Kanban (ou alors forme électronique) car pas de lieu dédié
- > L'absence de cadre (être full time) pousse à se disperser ... Mais avec un peu de discipline ça va
- > Les pratiques « Agiles » comme TDD / XP donnent un cadre qui évite la dispersion
- > L'expérience pousse à s'outiller (UDD, tests auto ...), ce qui influence le design de l'application

- > Beaucoup de nouvelles choses apparaissent qui ne sont pas répertoriées (fonc & tech)
- > Risques technos : comment on fait de l'IOC pour les tests ? Est-ce qu'on en fait si c'est possible ?
- > Risques fonc :
- > Risques méthodo :
 - + Difficile de faire un backlog : va se ressentir sur le tracking et l'appréciation de l'avancement → incapables de connaître notre vélocité

- > Story Points vs J/H

- > Kanban board
- > UDD
- > TDD / IOC
- > User Story
- > Story Points
- > ...

Quelques différences notables entre un cycle en 'V' et l'Agile ?

- > Chiffrage en j/h vs Story points
- > Spec en amont vs Spec par étapes
- > Recette en aval vs Recette en continu
- > Relation client/fournisseur vs Équipe projet

Objectif : S'adapter au changement pour produire de meilleurs logiciels

- > Pilotage à la valeur métier : Prioriser le développement (fabriquer le plus utile le plus tôt)
 - + Périmètre fonctionnel complété au fil des livraisons
 - + Si l'on s'arrête en cours, on a tiré le maximum de ce que l'on pouvait du budget / on a un logiciel qui fonctionne.

- > Livrer un logiciel qui correspond aux attentes
 - + Rapprocher le métier des développeurs
 - + Adapter le logiciel en fonction des nouveaux besoins

- > Un logiciel bien conçu
 - + Limiter le coût du défaut / l'apparition de défauts
 - + Rythme soutenable

- > Des démonstrations fréquentes
 - + Valider l'implémentation régulièrement (visibilité)
 - + Faire des projections réalistes sur quand le logiciel sera prêt
 - + « fail fast »

- > Un outil pour tout cela : la construction par itérations

Un peu de vocabulaire : ce qu'on trouve dans une itération

- > User Story :
 - + Élément fonctionnel assez petit pour être développé en une semaine ou deux
- > Démonstration
 - + Démonstration du logiciel au PO et/ou aux utilisateurs. C'est à ce moment que l'on valide les User Stories
- > Iteration
 - + Période de temps entre deux démonstrations
- > Planning game
 - + Se passe en début d'itération. Moment où l'on assigne une valeur en points à chaque US
- > Kanban
 - + Tableau où figurent les US en cours de développement. Permet de matérialiser l'avancement d'une itération, d'identifier des goulets d'étranglement opérationnels.
- > Stand Up Meeting
 - + Réunion d'équipe. Elle est courte et permet à chaque membre de l'équipe de partager ou il en est dans son travail, demander/proposer de l'aide

Un peu de vocabulaire : en dehors de l'itération

- > SCRUM
- > eXtreme Programming
- > Roadmap
 - + Liste des grandes fonctionnalités souhaitées pour le logiciel. Elles sont priorisées et réparties dans des silos fonctionnels
- > Backlog
 - + Liste des US restantes
- > Vélocité
 - + Capacité d'une équipe de développement à produire
- > Burndown

- > Méthodo
 - + SCRUM and XP from the trenches
 - + Becoming a technical leader
 - + Kanban: Successful Evolutionary Change for Your Technology Business
- > Dév / craftsmanship
 - + Clean Code (Robert C. Martin)