Génie Logiciel

Scrum & Méthodes agiles

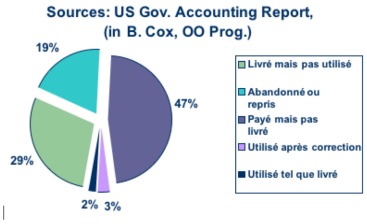
Historique

Evolution du processus de développement

origine

On code et on débogue..

⇨ Crise du logiciel



années 70-80

- Programmation structurée (Pascal, Ada, ..)
- & Cycles en cascade, en V

Objectifs

- Le client veut des garanties sur ce qu'il obtiendra en fin de projet
- Le chef de projet souhaite disposer des informations nécessaires à l'organisation de son équipe.

Le résultat

Puis..

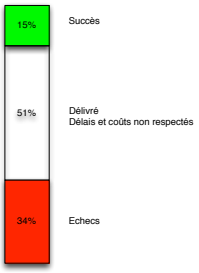
➔POO, Méthodes itératives (dont UP) ..

Malgré tout..

Taux d'échecs des projets informatiques

Rapport Chaos 003

Statistiques effectuées en 2004



Méthodes agiles

Les raisons de ces échecs (selon «agiles»)

- **La lourdeur**
  - Les méthodologies sont bureaucratiques
  - Le travail à fournir pour suivre la méthodologie entraîne de la lenteur dans l'avancement du projet

  ⇨ **«méthodologies lourdes»**
  «méthodologies monumentales»

- **Des lacunes**

  Certains facteurs ne sont pas assez pris en compte:

  - Les changements d'exigences
  - L'inflation des fonctionnalités

Méthodes agiles

Apparition des méthodes agiles

Dès années 2000 → Avec grande diffusion dans les entreprises

Quelques dates (rappel)
- années 90 ⇨ **POO** et début du **développement itératif**
- années 95-2000 ⇨ **UP**

Quelques méthodes agiles

- XP (E**X**treme **P**rogramming) ⇨ *La plus connue au départ*
- Scrum ⇨ *La plus utilisée aujourd'hui*

- Feature Driven Development
- ASD
- Etc.

Méthodes agiles

Les méthodes agiles s'appuient sur le «Software Manifesto» de 2001

- Séminaire de deux jours à Snowbird Utah en Février 2001

⇨ «Alliance Agile»
⇨ «**Software Manifesto**»

Extraits du Software Manifesto

«Individuals and interaction over processes and tools»

«Les méthodes agiles sont orientées vers l'humain, plutôt que vers le processus. Elles s'efforcent explicitement de travailler avec la nature de l'homme plutôt que contre elle, et de souligner que le développement informatique devrait être une activité agréable»

- **Optique traditionnelle**
  - les personnes sont remplaçables.
  - les gestionnaires manipulent non point des personnes mais des **ressources**
  - modèles de ressources: Analyste, Codeurs, Testeurs, Manager, etc..
  - les développeurs sont des **unités de programmation plug-and-play**
- **Optique «Agile»**
  - travail de développement = travail hautement créatif et professionnel
  - Effort = design ⇒ personnel créatif et talentueux
  - La motivation des développeurs est un facteur primordial à la réussite des projets

Extraits du Software Manifesto

«Working software over comprehensive documentation»

«Sur plusieurs aspects, les méthodes agiles sont plutôt orientées code: en suivant la voie selon laquelle l'élément clef de la documentation est le code. La différence la plus immédiate est qu'elles sont moins orientées paperasse, évoquant généralement une quantité moindre de documentation pour une tâche donnée»

⇒ Faire une spécification détaillée et compréhensible par les développeurs perd de son sens **si le client est à proximité**

P.e. avec XP «pur», la spécification du logiciel se résume à:

- scénarios (**user stories**)
- spécification des **tests de fonctionnalités**

Extraits du Software Manifesto

«Responding to change over following a plan»

«Les méthodes agiles sont adaptatives plutôt que prédictives. Les méthodes lourdes tendent à planifier de grosses parts du processus logiciel en détail pendant une durée importante, ce qui fonctionne correctement tant qu'il n'y a pas de changements. Donc, leur nature est de résister au changement. Les méthodes agiles, par contre, s'accommodent favorablement du changement. Elles s'efforcent d'être des processus qui s'adaptent au changement. Elles s'épanouissent dans le changement, au point de se changer elles-mêmes»

Car tout ne peut que changer..

Loi de Murphy - LEM»:

- Une chose est toujours plus compliquée qu'elle n'en a l'air
- Tout prend plus de temps qu'on ne le pense
- Tout ce qui peut aller mal ira mal

Comment répondre au changement ?

☺ faire évoluer progressivement le cahier des charges..

*est **plus avantageux** que*

☹ chercher à le spécifier et le concevoir complètement dès le départ

Extraits du Software Manifesto

Le processus de développement doit être lui-même adaptatif !

- L'équipe découvre ce qui fonctionne pour elle et le processus se modifie
- Un bilan au terme de chaque itération: qu'est-ce que l'on a réussi? qu'est-ce que l'on a appris? qu'est-ce que l'on peut améliorer?

⇨ Une méthodologie par équipe

⇨ Plus grande motivation des développeurs (préfèrent un processus adaptatif)

Méthodes agiles: Principes fondamentaux

- Suivent un mode de développement **itératif** et **incrémental**

  ⇒ Implication et collaboration et du client

- Une **planification de projet évolutive**
- Encouragent les **livraisons fréquentes** au client («releases»)
  (Itérations de 2 à 4 semaines)
- Proposent des pratiques qui encouragent **l'agilité** (souplesse d'adaptation) et la **réponse rapide aux changements**

⇔ UP avec souplesse++ et provocation changement

Principes fondamentaux Agile – Suite..

- Pour encourager l'agilité..
  - Une conception la plus simple possible au départ
  - Mais.. évolutive avec **refactoring** (remaniement de code)
  - Un côté moins bureaucratique
    - Moins orientées paperasse
    - Plus orientées code
- Programmation en binôme (méthode XP)
- **D**éveloppement **D**irigé par les **T**ests

  (→ **T**est **D**riven **D**evelopment)

Ca n'est plus vraiment UP..

Sur quels types de projets utiliser une méthode Agile?

Petite équipe (10-15)

- L'équipe doit être composée d'une majorité de seniors
- Qui maîtrisent la gestion de projet
- Qui sont responsables et motivés

Très adapté si exigences floues et variables

Client disponible

Exemple XP: Le client est **partie prenante**, responsable de :

- La spécification des scénarios & des tests de fonctionnalités
- Le pilotage des itérations

Sur quels types de projets utiliser une méthode Agile?

- *Petite équipe (10-15)*
- *Exigences floues et variables*
- *Client disponible*
- *Client partie prenante*

Agile ne s'applique pas à TOUS les TYPES de PROJETS !!!

Agile n'est pas à mettre entre toutes les mains..

UP est-elle agile?

UP peut s'accommoder de concepts Agiles

P.e., Craig Larman incite à utiliser UP dans sa forme agile, c'est-à-dire légère:

- Planification des itérations effectuées avec la **totalité de l'équipe** *(Partage des responsabilités)*
- Programmation en binôme

☹ **Toutefois, le côté très formel d'UP n'est pas considéré comme «agile»**

- Description précise des artefacts qui devront être élaborés par les développeurs
- Rôle assez rigide joué par les membres de l'équipe de développement
- Découpage en 4 phases (initialisation, élaboration, construction et transition) avec objectifs très précis: *Les méthodes agiles ne prévoient pas de phases à priori!*
- La philosophie **T**est **D**riven **D**evelopment n'est pas mise en avant

UP est-elle agile ? - *Suite*

UP s'appuie sur 3 fondements

- Piloté par les cas d'utilisation = **Agile**
- Itératif et incrémental = **Agile**
- Centré sur l'architecture ≠ **Agile**

UP est centré sur l'architecture

Architecture

- La couche **infrastructure**

  Exemple: couche client-serveur d'une application WEB

  - souvent indépendante de la partie métier
  - produit direct de la culture de l'entreprise et de ses compétences

- La couche **métier**

Objectif de UP: minimiser les risques d'échecs

- On recherche la stabilité
- Les principaux éléments de **l'architecture** sont spécifiés, voire réalisés, **pendant la phase d'élaboration**

Agiles: Architecture élaborée au fur et à mesure du développement