Management agile - Méthodes agiles

Nuwan Herath

2022-2023

Ce cours se base aussi sur une intervention de Pierre Auclair et le cours de Claude Godart

Les méthodes agiles

L'agilité

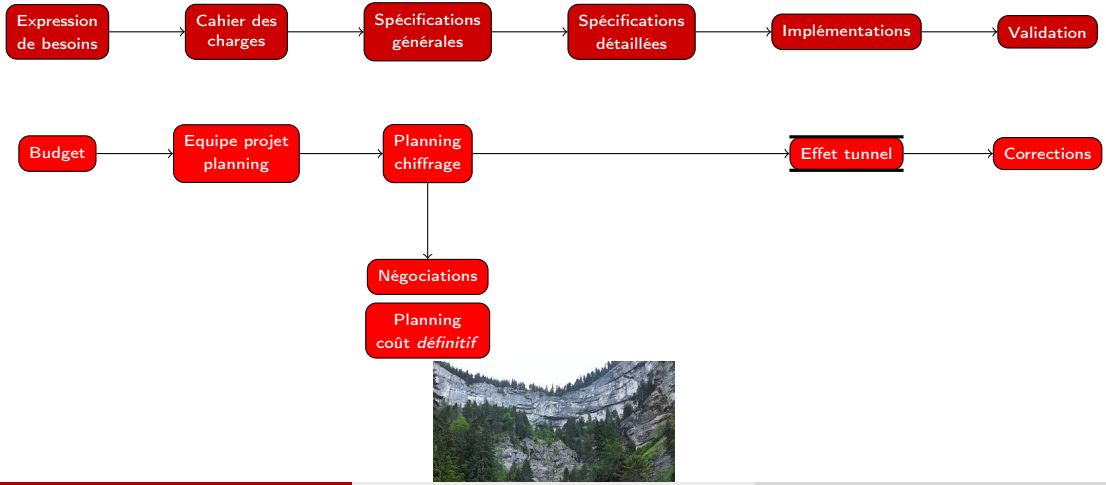
Au sens de l'automatique ("étude des systèmes dynamiques"), l'agilité est un asservissement

Le principe de base d'un asservissement est de mesurer, en permanence, l'écart entre la valeur réelle de la grandeur à asservir et la valeur de consigne que l'on désire atteindre, et de calculer la commande appropriée à appliquer à un (ou des) actionneur(s) de façon à réduire cet écart le plus rapidement possible[1]

[1] Wikipédia, Asservissement

Ce qu'on veut éviter

Projet "standard"



Le Manifeste pour le développement agile de logiciels, février 2001 (1/3)

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Le Manifeste pour le développement agile de logiciels, février 2001 (2/3)

We follow these principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Le Manifeste pour le développement agile de logiciels, février 2001 (3/3)

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity – the art of maximizing the amount of work not done – is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Traduction du manifeste

Voir `http://agilemanifesto.org/iso/fr/manifesto.html`

Parmi les signataires

Kent Beck	extreme programming
Alistair Cockburn	Crystal clear
Ward Cunning	Wiki
Martin Fowler	Dynamic System Development Method (DSDM)
Robert Cecil Martin	Clean Architecture
Ken Schwaber	Scrum
Jeff Sutherland	Scrum
Dave Thomas	DSDM

Quelques caractéristiques

Les pratiques agiles sont

- itératives (limitation temporelle des itérations)
- incrémentales
- adaptatives

Vision agile

Source : Thomas Quine, CC-BY 2.0

- Cycles courts et contraints
- Communications courtes ET fréquentes
- Intégration du client dans l'équipe

Quelques exemples de méthodes agiles

- Lean management
- Scrum
- eXtreme Programming
- Kanban

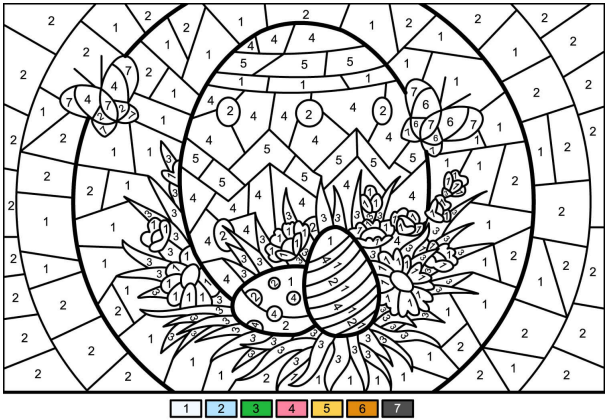
Une parenthèse

Complexifier progressivement le produit

- Créer une version démontrable
- Itérer à partir de cette version
- Vérifier que cela correspond au besoin utilisateur
- Respecter la vision à long terme

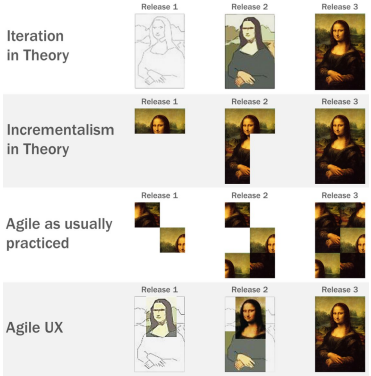
Une métaphore

Dessiner ce n'est pas faire un coloriage numéroté



L'expérience utilisateur de l'agilité

Cf. l'article du blog de Jeff Patton : Don't Know What I Want, But I Know How to Get It



Source : `https://twitter.com/CGLambdin/status/712713187833958400`

La crainte du développeur

- "On sait ce qu'on veut. Combien de temps est-ce que vous pensez que ça va vous prendre ?"
- "On a besoin d'écrire un cahier des charges détaillé avant de commencer le développement."

$\rightarrow$ mauvaise compréhension de ce qu'est une "itération" dans les pratiques agiles

Pourquoi itérer ?

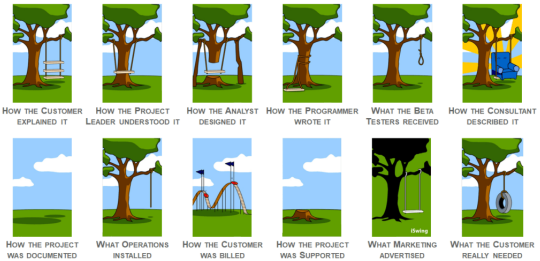
- Pour converger vers la bonne solution / une solution satisfaisante
- Améliorer la solution candidate actuelle

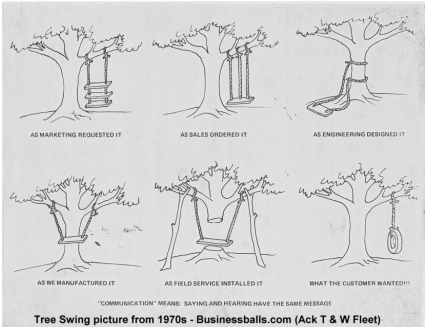
Pourquoi incrémenter ?

- Pour construire des fonctionnalités petit à petit (avoir quelque chose à déployer)
- Pour avoir des retours des utilisateurs le plus tôt possible

$\rightarrow$ Produire ce dont le client *a besoin*, pas ce qu'il *veut*

La balançoire

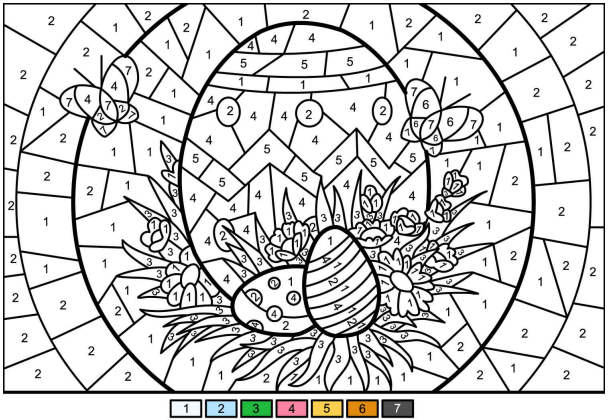




Source : `https://www.action-engineering.com/blog/agile-acceptance-criteria/`

Une métaphore

Dessiner ce n'est pas faire un coloriage numéroté



Le changement

L'inévitable changement

Le changement est **inévitable** dans les gros projets logiciels

- pressions extérieures
- changement des priorités
- compétition
- nouvelles technologies
- nouvellles approches de conception

Surcoût de développement logiciel

Le prototypage, l'anticipation du changement

Le prototype est une version initiale du système servant à

- prouver un concept
- essayer des options
- mieux comprendre le problème
- ...

La livraison incrémentale, la tolérance au changement

Avantages

- De premiers incréments servant de prototypes, mais déjà des parties du système final (pas de réapprentissage)
- Utilisable dès le début
- Facile d'introduire des modification
- Moins de risques de problèmes critiques

La livraison incrémentale, la tolérance au changement

Inconvénients

- Pas pratique quand on conçoit une solution de remplacement pour un système déjà existant
- Difficile d'anticiper toutes les briques de bases pour tous les incréments à venir
- Difficile d'écrire un contrat sans spécification complète

Scrum

Qu'est-ce que Scrum ?

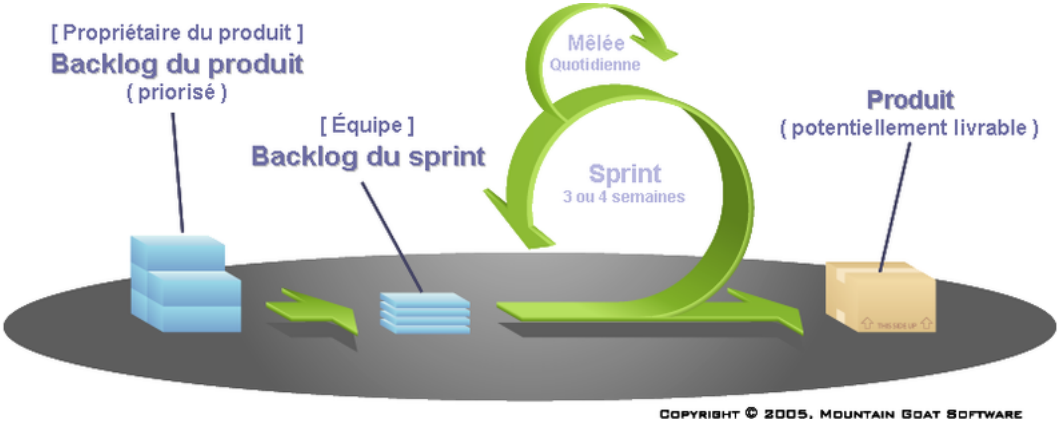
Scrum est un cadre de développement de produits complexes
Scrum est centrée sur l'idée d'une **équipe soudée** avec un **but à atteindre**

Source : Zegreg63, CC-BY-SA 3.0

On le doit à Ken Schwaber et Jeff Sutherland
scrum : mêlée

Scrum en une image

L'agilité est un asservissement (au sens de l'automatique) !



Source : Wikimedia Commons

Deux exemples de pratiques

TDD

Test Driven Development

- Créer un jeu de tests
- Vérifier qu'ils ne passent pas
- Ecrire le code
- Vérifier que l'ensemble des tests passent
- Réusiner / *refactoriser* le code

Pair programming

2 personnes et 1 seul clavier
20 min chacun
ou
l'un écrit les tests, l'autre les fonctions, puis les rôles s'inversent et ainsi de suite

Source : Vincent Déniel, CC-BY-NC 4.0

Conclusion

Trois aphorismes

- Ideas move faster than documents
- There is more to owning software than just paying for it
- Measure twice cut once saves wood, but software isn't made of wood

Source : `http://www.agile-process.org/proverbs.html`

Résumé de l'agilité

L'agilité c'est :

- une équipe responsable et autonome
- un boucle de rétroaction / *feedback loop*
- une pratique, mais pas un dogme

Quand choisir l'agilité ?

- Processus orienté plan
  - Besoin d'une spécification claire, d'une conception avant de démarrer la programmation
  - Projet de taille importante avec une grosse équipe de développement
  - Système complexe, critique nécessitant une analyse importante
  - Logiciel appelé à une longue vie et à une maintenance importante
  - Equipe distribuée et/ou en partie externalisée
  - Système soumis à une régulation importante
- Processus agiles
  - Stratégie incrémentale avec l'implication forte des utilisateurs
  - Equipe de taille réduite avec une bonne expérience
  - Possibilité de s'appuyer sur un outillage efficace