Cycle de vie du logiciel

...

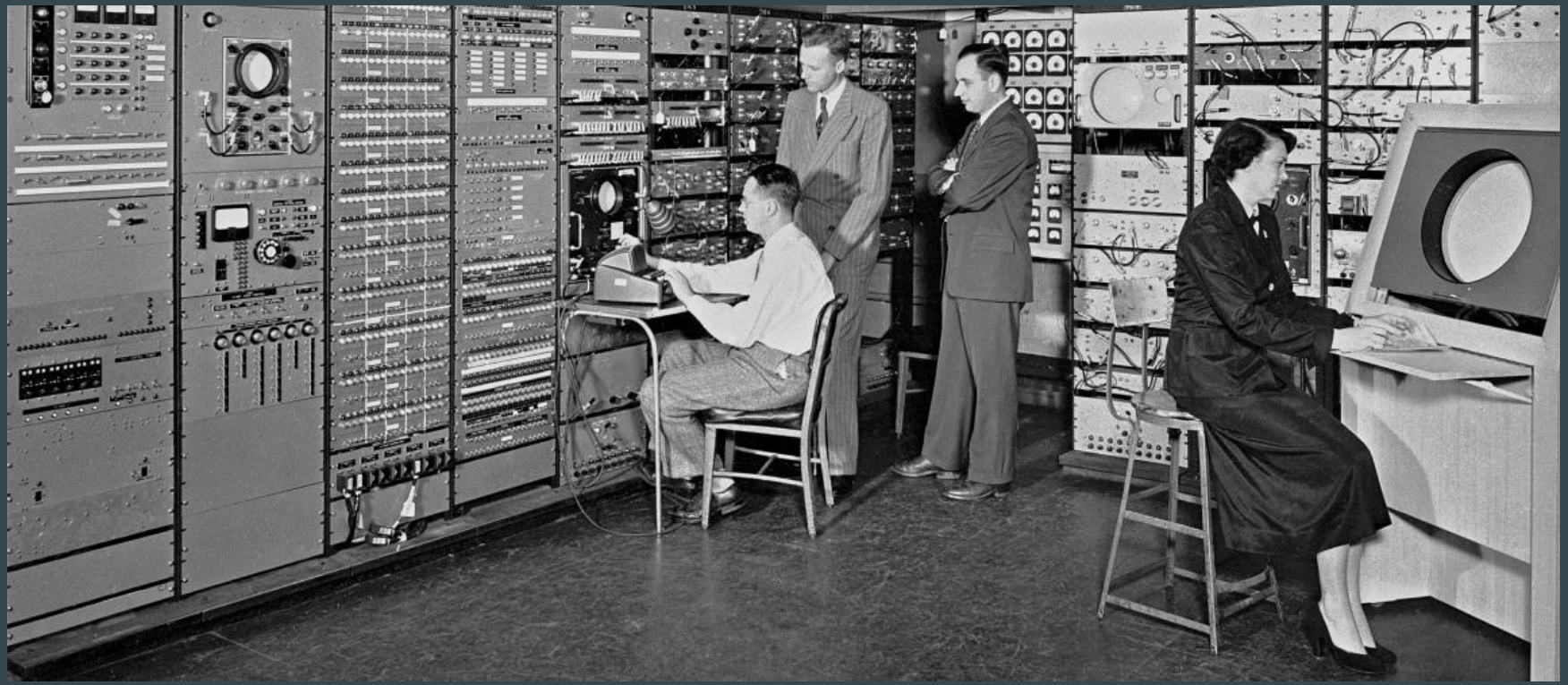
De la construction de bâtiment au déploiement continue

Avez-vous déjà entendu, dit ou pensé...

- Livrer c'est coûteux, c'est risqué donc je préfère ne pas le faire souvent
- Pour la qualité, on a planifié une revue qualité à la fin du projet
- Tester c'est long alors pour gagner du temps on testera à la fin quand tout sera fini
- Rien ne marche encore mais d'après le planning et le RAF, nous sommes à 80% d'avancement
- Tout est fait, y'a plus qu'à tester!

http://www.excusesdedev.com/

Développe t-on encore comme cela?

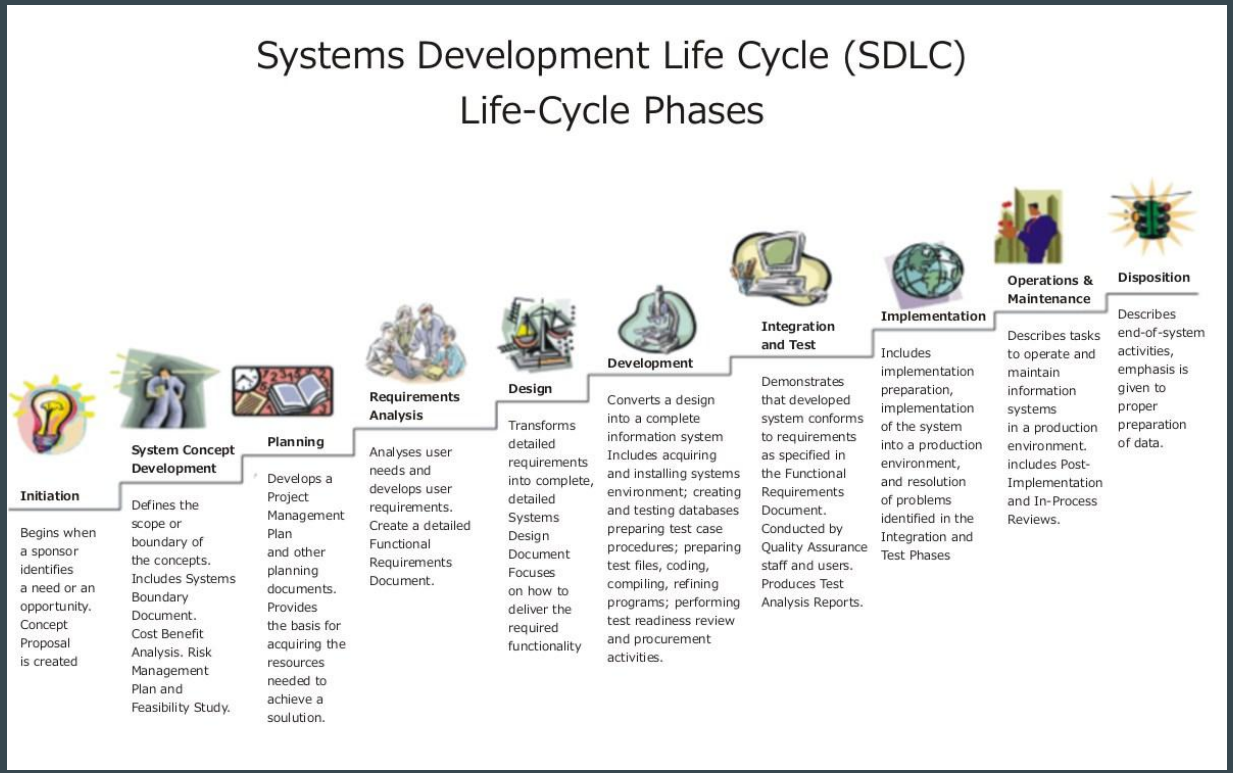


http://www.mitre.org/

Ou comme cela?

http://bloggelidaza.blogspot.fr

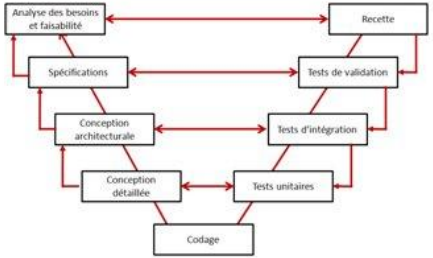
SDLC



https://en.wikipedia.org/wiki/Systems_development_life_cycle

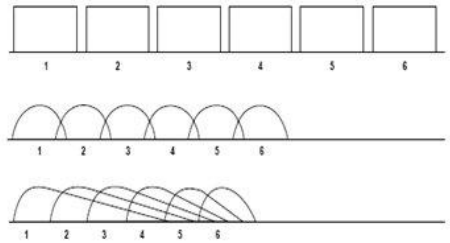
Une synthèse des différents cycles de développement

Cascade



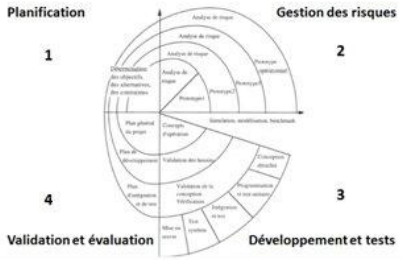
Cycle en V

Incrémental et Concourant



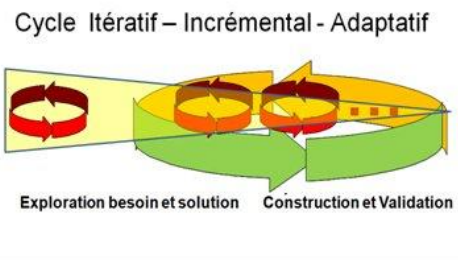
New New Product Development Game

Itératif et Incrémental



La Spirale

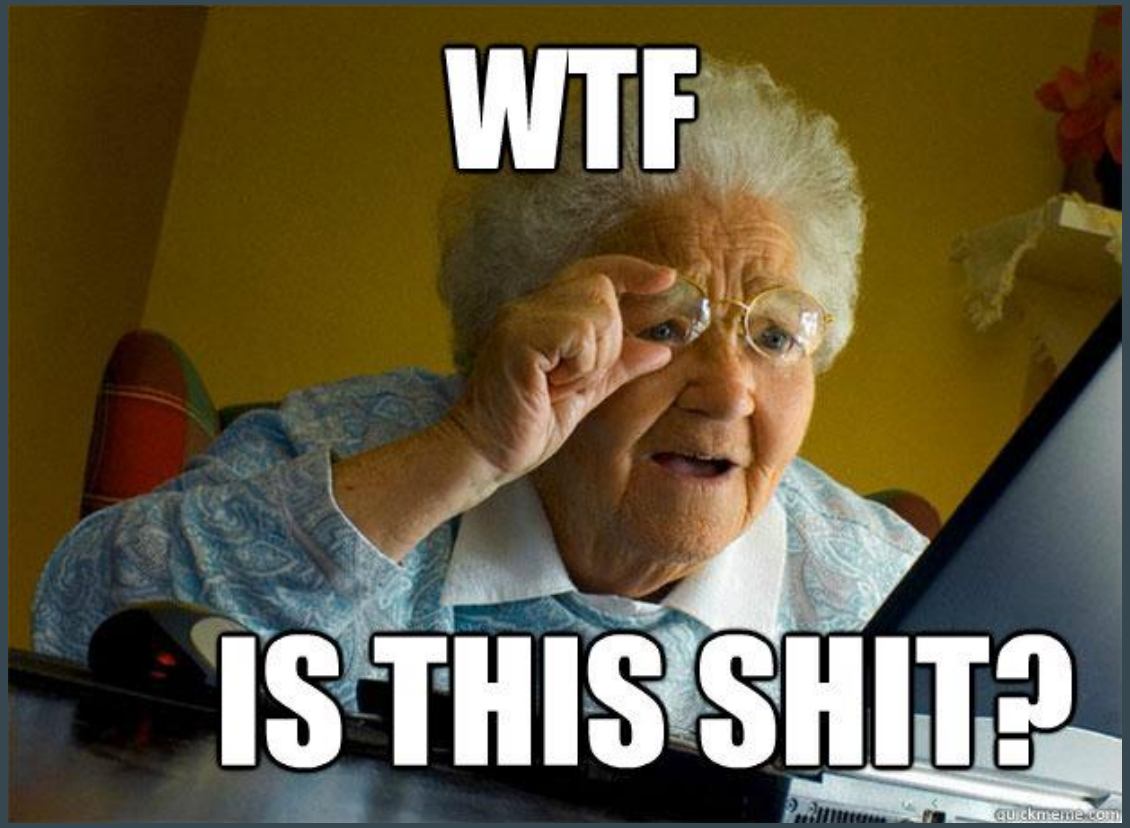
Itératif Incrémental Adaptatif



Agile : RAD, XP, Scrum, PUMA

http://www.entreprise-agile.com/HistoAgile.pdf

Effet tunnel



Les bonnes pratiques changent

Par exemple, les architectures :

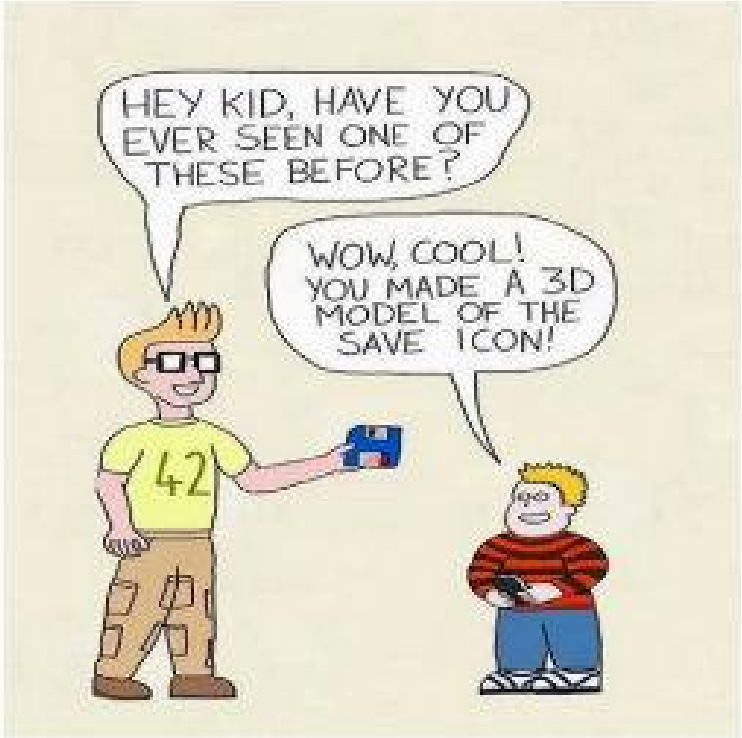
- Client-serveur : la logique métier est plutôt côté client, la persistance côté serveur
- Architectures n-tiers : le côté client ne gère que la présentation, toute la logique est côté serveur qui distingue une partie applicative ou métier, d'une partie persistance
- SOFEA : on remonte à nouveau la logique métier côté client, plusieurs modules ou services distincts sont invoqués par le client

Les bonnes pratiques changent

Autre exemple, la persistance :

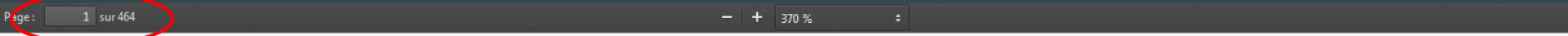
- Dénormalisation en plusieurs fichiers : articles, commandes, produits, factures... avec de la duplication
- Relationnel : normalisation, relations/références entre les données, duplication proscrite
- Retour à la dénormalisation : NoSQL, moteur de recherche, cache applicatif

Les temps changent...



https://dmnphg4ua2zej.cloudfront.net

Installation



Installation and Upgrade Guide

Notre métier : sans cesse s'adapter aux évolutions

Le contexte actuel

- Livrer vite (ex. Google et les betas), livrer souvent, intéragir avec l'utilisateur, recueillir du feedback, affiner, améliorer, relivrer, re-recueillir feedback etc... toutes les 2, 3 ou 4 semaines
- On donne une version incomplète à l'utilisateur mais fonctionnelle. Il n'y a pas toutes les fonctionnalités, mais on apporte de la valeur, on prend en compte ses retours. Il nous confirme si c'est bien cela qu'il voulait, puis nous priorise les prochaines choses à faire.
- Chaque développement est vu comme une évolution de la version précédente. La capacité à faire évoluer du code déjà en production est aujourd'hui essentielle.
- Si tester prend 10 jours, packager 1 jour et livrer 2 jours, c'est ingérable. Les outils d'automatisation ont émergés pour résoudre ce problème, fiabiliser et accélérer les livraisons

Contexte utilisateur

- Nos utilisateurs utilisent l'informatique en dehors de leur entreprise
- Ils utilisent leur PC, tablettes, smartphones tous les jours
- Ils installent des logiciels en quelques clics, peuvent s'en plaindre sur les forums, les boutiques ou sur le site de l'éditeur
- Ils utilisent des sites et applications qui évoluent sans cesse, dont l'ergonomie et l' utilisabilité sont très travaillés
- Ils n'ent sont que plus exigeants quant aux logiciels d'entreprise...

Illustration des changements : cas facebook

- Tous les tests sont automatisés, unitaires, intégration, acceptation
- Lors d'un commit, une version est construite, testée, déployée sur un serveur de tests
- Un robot, dans un tchat, communique avec le développeur, indiquant l'URL du serveur de test. Le développeur peut alors vérifier si tout est OK
- Le robot lui propose de released une version et de la pousser en production
- Si le développeur valide, la modification est poussée sur un des serveurs de production, la réplication sur les autres noeuds de la ferme de serveur se fait par Bit Torrent

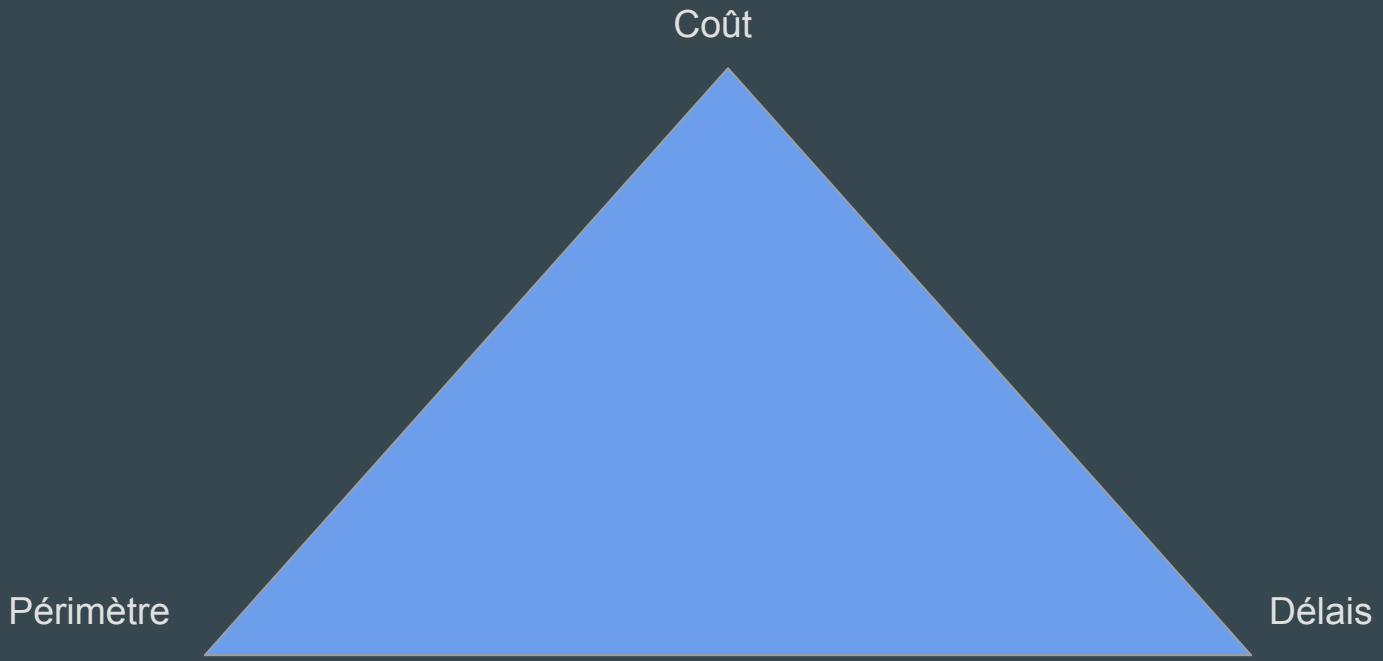
https://vimeo.com/56362484
http://www.infoq.com/presentations/Facebook-Release-Process

12 questions pour améliorer ses développements

1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. Do you have a bug database?
5. Do you fix bugs before writing new code?
6. Do you have an up-to-date schedule?
7. Do you have a spec?
8. Do programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?

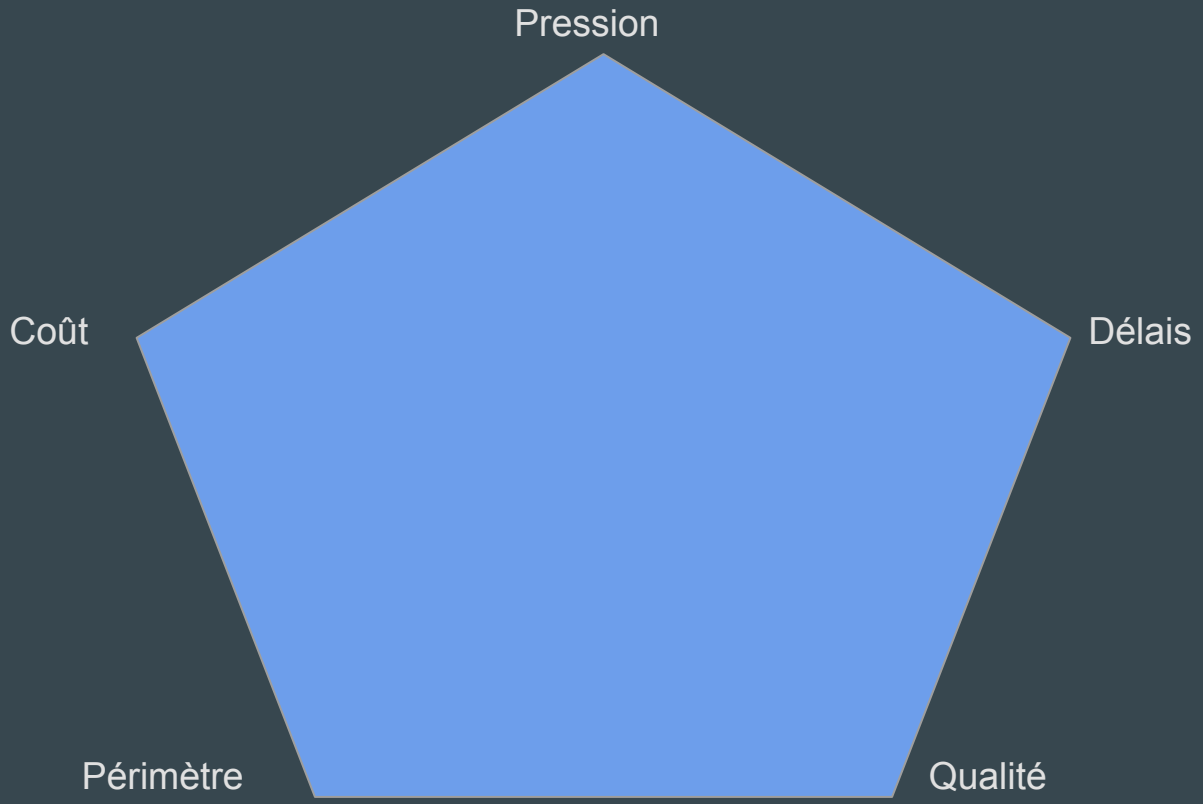
http://www.joelonsoftware.com/articles/fog0000000043.html

Le triangle magique...

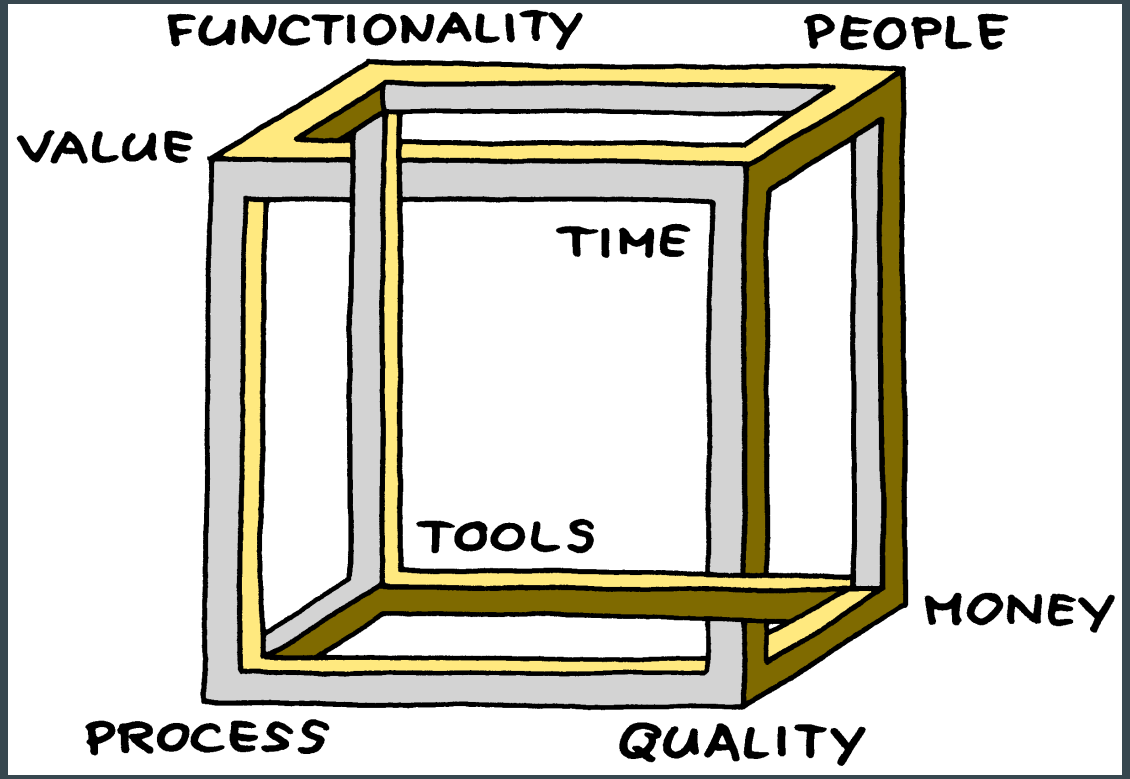


… ou le carré?

ou l'hexagone?



ou 7 dimensions



Jurgen Appelo, Management 3.0, 2011

Les 4 valeurs agiles

Synthèse

- Un projet informatique, ce n’est pas uniquement du code
- Les méthodes et les pratiques doivent s’adapter aux contextes et aux évolutions technologiques
- Penser **produit** et pas seulement **projet**
- La vie d’un produit logiciel commence avec l’idée du logiciel et s’arrête à sa désinstallation et/ou son obsolescence
- Dans la vie du logicielle, on trouve beaucoup d’activités répétitives et automatisables

12 principes agiles

- Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
- Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.
- Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
- Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
- Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.

12 principes agiles

- La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.
- Un logiciel opérationnel est la principale mesure d'avancement.
- Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
- Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.
- La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.
- Les meilleures architectures, spécifications et conceptions émergent d'équipes autoorganisées.
- À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.