Cours de Développement Web
Processus de développement d'un logiciel

Frédéric Flouvat

Université de la Nouvelle-Calédonie

frederic.flouvat@univ-nc.nc

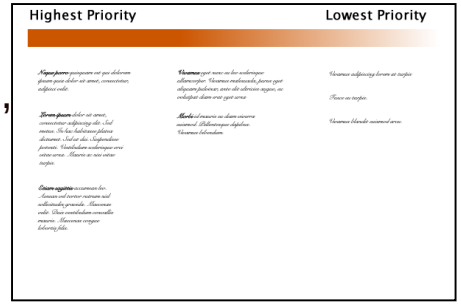
Quelques références bibliographiques

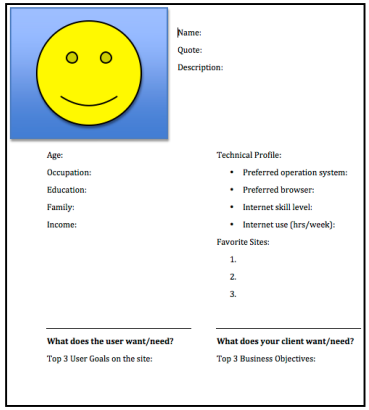
- "CS 5150 Software Engineering", W.Y. Arms, Cornell University, http://www.cs.cornell.edu/courses/cs5150/2018sp/lectures.html
- "Software development life cycle", Tutorials point, https://www.tutorialspoint.com/sdlc/sdlc_quick_guide.htm
- "Software development life cycle", Movistar – GSL MIT, http://gsl.mit.edu/media/programs/peru-summer-2014/materials/t04-_software_development_life_cycle.pdf
- "The SDLC: 7 phases, popular models, benefits & more", D. Swersky, https://raygun.com/blog/software-development-cycle/
- "Manifeste pour le développement Agile de logiciel", http://agilemanifesto.org/iso/fr/principles.html
- "Agile Web Development – a Comprehensive Overview", Cody Arsenault, https://www.keycdn.com/blog/agile-web-development/
- "Thoughts on Design and Agile", M. E. Miller, Stanford Web Services Blog, https://swsblog.stanford.edu/blog/thoughts-design-and-agile
- "La méthodologie Scrum expliquée simplement – Le Guide Ultime de la Méthode Agile Scrum", S. Boyer, https://www.nutcache.com/fr/blog/methodologie-scrum/
- "Agile in a nutshell", http://www.agilenutshell.com
- "Agile Modeling Home Page", http://agilemodeling.com

Les différentes étapes de la vie d'un logiciel

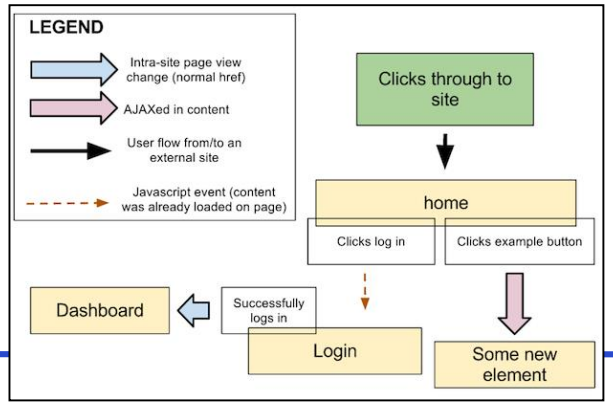
- Etude de faisabilité et planification
  - Objectifs du projet ? réalisabilité technique ? coûts ? calendrier ? ressources ? risques ? ➔ décision

- Définition des besoins clients
  - Identification des profils des utilisateurs (*personas*), recensement des besoins (fonctionnalités, performances, matériels, organisation, etc.), définition de scénarios/cas d'utilisation



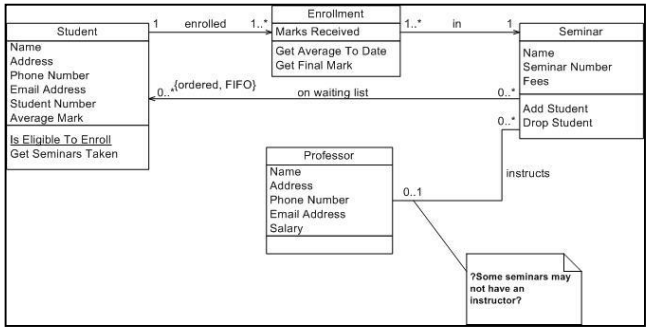


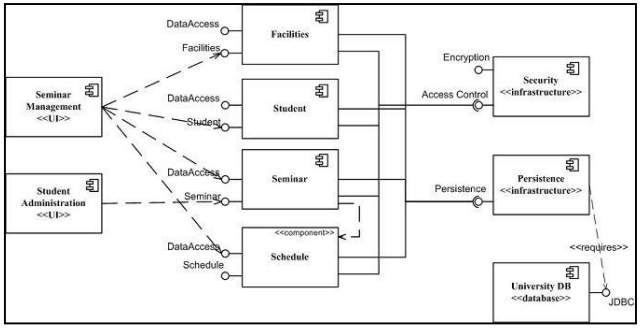
- Design de l'interface utilisateur
  - Diagrammes d'expérience utilisateur (*UX flow charts*), squelettes graphiques (*wireframes*), maquettes graphiques, prototypes, tests et validations clients



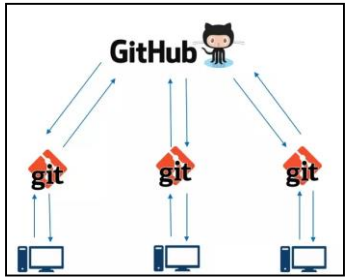
Les différentes étapes de la vie d'un logiciel

- Design de l'architecture du logiciel
  - Organisation des données (diagramme E/A), architecture du code (diagramme de classes), architecture physique et réseau (diagrammes de composants), sécurité, technologies et outils utilisés pour le développement





- Programmation du logiciel
  - Coder, tester (à automatiser si possible), partager et faire valider le code à l'équipe

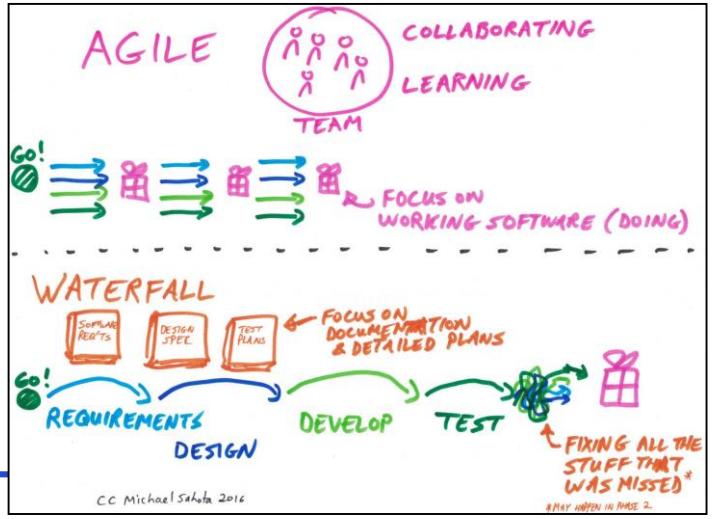


- Test utilisateur du logiciel et déploiement
  - Test complet par les clients et comparaison aux besoins, livraison et mise en production

- Utilisation et maintenance

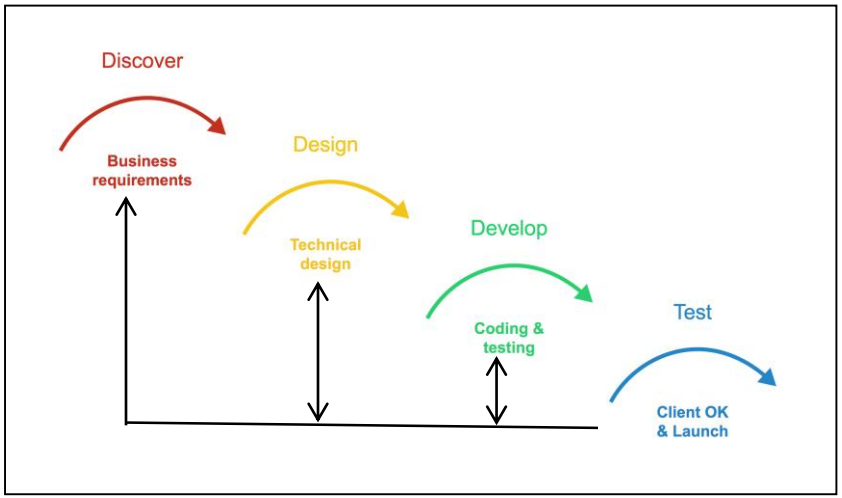
Les modèles de développement logiciel

- Différentes façons de s'organiser autour de ces étapes
- Différents modèles de développement en fonction des projets et des équipes
  - Chacun à des avantages et des inconvénients
  - Mais toujours les mêmes objectifs
    - avoir un bon logiciel (fonctionnalités, utilisabilité, maintenabilité, efficacité, etc.)
    - limiter les risques (ne répond pas aux besoins, coût trop élevé, durée de développement trop longue, etc.)
    - faciliter le suivi du projet par le responsable
    - favoriser le travail en équipe
- Exemples de processus de développement logiciel
  - **Modèle en cascade**
  - Modèle itératif
  - Modèle en spiral
  - Modèle en V
  - **Modèle Agile**
  - …



Le modèle en cascade (*waterfall*)

- Un développement séquentiel / linéaire
  - Finir chaque étape avant de commencer la suivante
  - Possibilité de revenir en arrière en fonction des retours clients à la fin de chaque étape

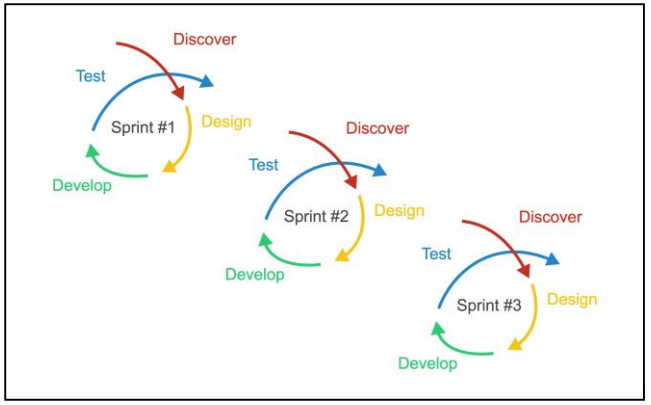


- Une approche adaptée si
  - Des besoins fixes, clairs et bien documentés
  - Une technologie maîtrisée

- Avantages & Inconvénients
  - \+ Simple à comprendre / utiliser
  - \+ Facile à manager
  - \+ Des étapes clairement définies
  - \+ Un processus et des résultats bien documentés
  - \- Pas de logiciel opérationnel avant les dernières étapes du processus
  - \- Beaucoup de risques et d'incertitudes
  - \- Difficile d'évaluer l'avancement
  - \- Difficile d'intégrer de nouveaux besoins

L'approche Agile

- Un développement adaptatif, itératif et incrémental
  - Projet découpé en petites tâches (*sprints*)
  - Un *sprint* = toutes les étapes du développement logiciel
  - A chaque fin de *sprint*, une version utilisable du logiciel présentée aux clients



- Une approche adaptée si
  - Des besoins pouvant changer
  - Une équipe localisée à un seul endroit
  - Des clients / utilisateurs avec une bonne disponibilité

- Avantages & Inconvénients

+ Flexible
+ Met en avant la satisfaction client et le travail d'équipe
+ Des livraisons régulières de logiciels opérationnels
+ Des ressources et des coûts limités

- Dépend beaucoup des interactions avec les clients
- Importance de l'expérience de l'équipe
- Risque de retravailler continuellement le code et les besoins
- Peu de documentation

Le manifeste pour un développement Agile

- Agile = avant tout des principes, un condensé de bonnes pratiques
  http://agilemanifesto.org
  - Priorité à la satisfaction client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée
  - Accepter les changements même tard dans le projet
  - Livrer fréquemment un logiciel opérationnel avec des cycles de quelques semaines
  - Travailler avec les utilisateurs quotidiennement tout au long du projet
  - Motiver l'équipe de développement, lui fournir du soutien, et lui faire confiance
  - Favoriser le dialogue en face à face
  - Un logiciel opérationnel comme principale mesure d'avancement
  - Encourager un rythme de travail soutenable
  - Focus sur l'excellence technique et une bonne conception
  - Importance de la simplicité (minimiser le travail inutile)
  - Mettre en place des équipes s'auto-organisant
  - Essayer continuellement de trouver des moyens de devenir plus efficace et modifier son comportement

Méthodologies Agiles

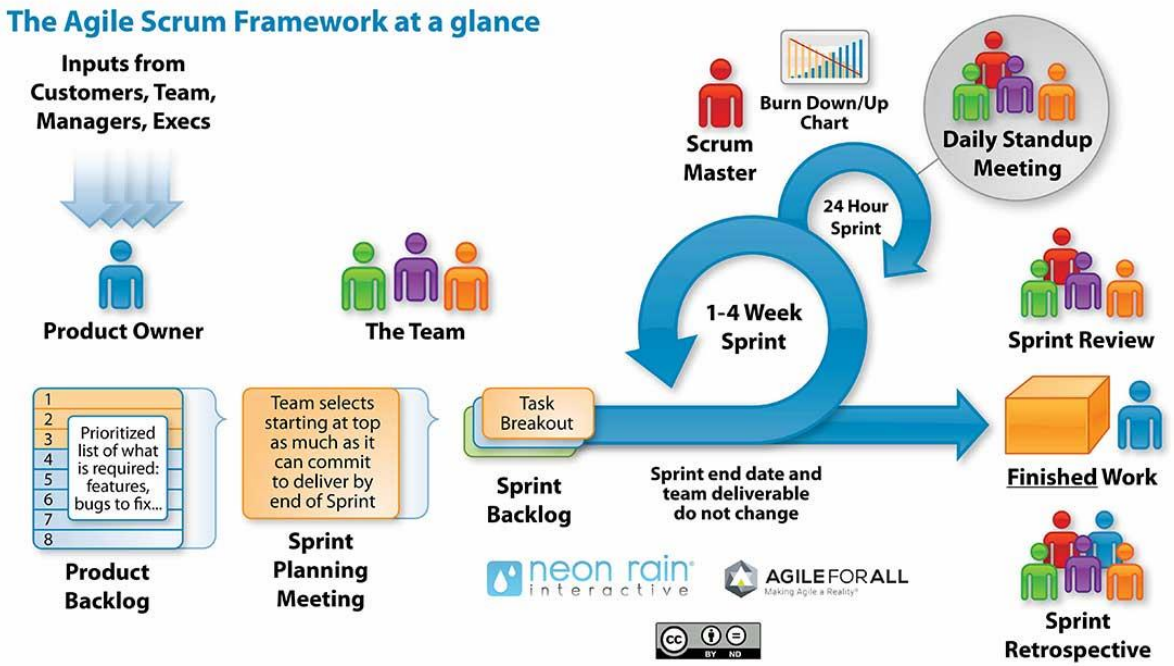
- Plus d'une vingtaine de méthodes dites "Agiles": RAD, XP, RUP, Kanban, LeSS, Scrum, …
- La plus connue: **Scrum**

Product Backlog: liste des fonctionnalités (avec des priorités)

Sprint Backlog: liste des tâches à accomplir pendant un sprint (construit à partir du *Product Backlog*)

Scrum Meetings: réunion journalière, courte, pour analyser ce qui a été fait et planifier la suite

Scrum Master: personne chargée d'organiser les réunions et la communications entre les membres de l'équipe



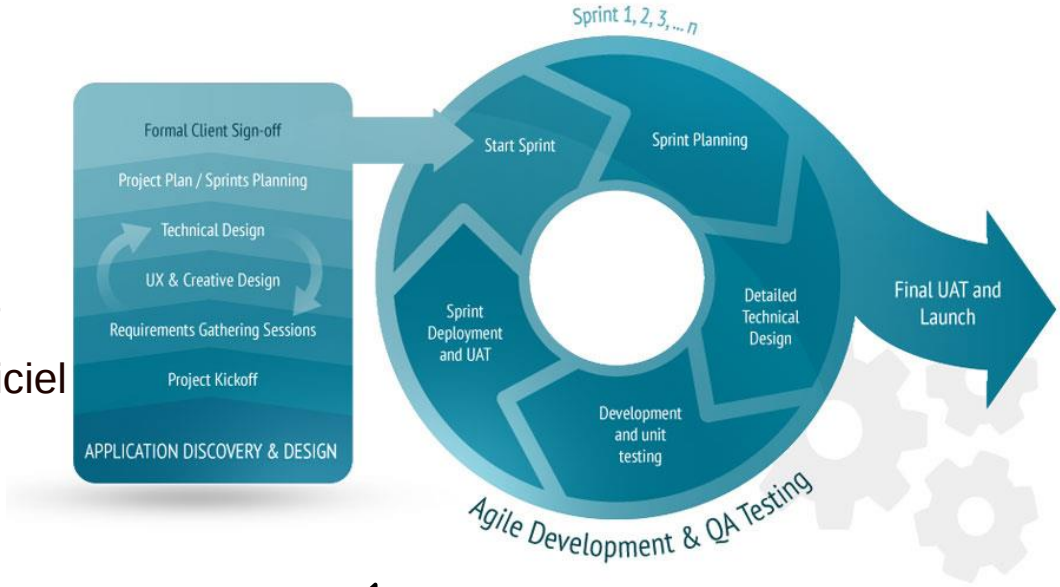
Idéalement, chaque sprint aboutit à une application totalement opérationnelle

Les processus hybrides

- En réalité, souvent plusieurs modèles combinés
- Par exemple, Cascade + Agile

Cascade (description haut-niveau)

- Etude de faisabilité
- Définition des besoins clients
- Design de l'interface utilisateur
- Design de l'architecture du logiciel
- Planification des sprints



Agile (détails)

- Planification détaillée du sprint courant
- Design détaillé de l'interface utilisateur
- Design détaillé de l'architecture du logiciel
- Programmation
- Test utilisateur du logiciel (UAT) et déploiement

sprint suivant