Rappels

Génie logiciels

GL = {concept, méthodes, techniques, outils}

- pour coordonner le développement logiciel
- pour répondre au CQFD (Cout, Qualité, Fonctionnalités et Délais)

Grady Booch (1991), Ingénierie du logiciel avec ADA

un développeur :

- demande beaucoup de capacités intellectuelle
- scientifique
- artiste

commenter le code bande de tâche.

Alistair Cockburn (10km brûlé)

qui serait le développement logiciel si on ne le considerait pas comme un développement logiciel

=> jeu coopératif, d'invention et de communication + travail collaboratif (sa réponse)

jeu:

- amusement
- règles
- compétition

développement logiciel = construction maison

- architecte → plan (conception)
- constructeur → instance (réalisation)

/!\ le client doit être satisfait et non pas seulement le développeur. /!\

- les besoins du client risquent d'évoluer en cours de développement
- fonctionnalités qu'il a demandé
- pas de bug / crash
- performant
- erreurs : compréhensibles

Méthode Agile

Introduction

agile → agilité → souplesse, l'adaptabilité au changement
but : trancher avec la lourdeur des méthodes classiques ou traditionnelles (80')
ASD : Adaptative Software Development
Crystal
Scrum
FDD : Feature Driven Development
DSDM
AM Agile Modeling
XP eXtreme Programming

Agile Manifest => http://agilemanifesto.org/iso/fr/
- Our highest priority is to satisfy the customer through early delivery of valuable software
    -développement itératif
- Deliver working software frequently for a couple of weeks to a couple of months with a preference for 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 their needs and trust them to get the job done (courage)
- the most efficient and effective method of conveying information to and within a development team is face-to-face conversation (communiquer !)

Le développement Agile se focalise sur les points suivant :

Agiles	Traditionnelles
Individus et itération	processus et outils
logiciel qui fonctionne	document compréhensible
collaboration client	négociation d'un contrat
Réponse au changement	plan déterminé

Extreme Programming XP

1) Problèmes liés au développement logiciel

- délais non respectés
- besoins mal identifés ou mal compris
- bugs lors de l'utilisation du logiciel

Répartition du travail ?

Model	**View**	**Controller**
Client		
Livre		

- Tâches -> répartition programmeurs

2.3.2) Kent Beck le père d'XP

Kent Beck et Ward Cunningham ont travaillé sur le projet C3 Chrysler Comprehensive Compensation.

L'ancien système = 2000 classes Smalltalk, 30000 méthodes, 18 mois de développement, million de $. Le projet débute en 1996 -> 1997 développement d'une méthodologie XP.

2.4) Fondements

- client est partie prenante du projet
- livraison rapide d'une première version du logiciel, puis versions successives feedback client
- auto organisation de l'équipe de développement
- test automatiques
- refactoring = amélioration constante du code

2.4.1) Démarches

Simplifier le plus possible les démarches
-> limiter le temps non productif

2.4.2) Modification des principes

Démarche adaptative plutôt que prédictive
L'XP tente d'intégrer le changement et d'en faire quelque chose de positif

2.5) La méthode

Conception	Réalisation	Livraison

En développement itératif on répète ces 3 tâches, à chaque livraisons on contacte le client.

2.5.1) Phase initiale / Phase d'exploration

rencontrer le client, comprendre ses besoins en terme de fonctionnalité
On réalise avec le client des user-stories (scénarios)
Se retrouver avec le client et définir les besoins sur une feuille bristol A5
plusieurs fonctionnalités si trop compliqué

2.5.2) Phase de planification

Le client présente les scénarios à l'équipe de développement.
Les développeurs vont tenter d'estimer la charge de développement d'un scénario (coût en nombre de points)
L'équipe estime sa vélocité (= nombres de points réalisés en une itération)

2.5.3) Travail en équipe

- définition et partage des tâches
  au début de chaque itération
    - faire le point sur l'itération précédente
    - organisation des scénarios
- responsabilité collective
  faire de son mieux lors du développement pour garder la qualité du logiciel
- pair programming (binôme)
  changement toutes les heures entre celui qui programme et celui qui vérifie mais aussi changement de binôme tous les jours
- stand up meeting
  tous les jours avant de commencer à travailler -> faire le point sur le jour précédent
  Quelques membres peuvent se réunir pour approfondir certains points

Références :

Dans ma boite de stage on était entrain de mettre en place Scrum ou Kanban en méthode Agile. On est parti à la fin vers Scrumban. C'est sympas à lire pour la gestion de projet :)

Scrum : https://www.scrum.org/

Scrum Kanban : http://www.crisp.se/file-uploads/Kanban-vs-Scrum.pdf

Scrumban : http://leansoftwareengineering.com/ksse/scrum-ban/