Processus de conception de SI
3/3 – Méthodes Agile

Yannick Prié
UFR Informatique – Université Claude Bernard Lyon 1
M1 MIAGE – SIMA / M1 Informatique MIF17
2008-2009

Plan du cours global

- 1/3 – Méthodes et processus
- 2/3 – Processus unifié
- **3/3 – Méthodes Agile**

Plan de ce cours

- Principes des méthodes Agile
- XP : eXtreme Programming
- Scrum
- Autres méthodes

Petite histoire des méthodes Agile

- Années 90
  - réaction contre les grosses méthodes
  - prise en compte de facteurs liés au développement logiciel
- Fin années 90
  - méthodes
    - d'abord des pratiques liées à des consultants, puis des livres
    - XP, Scrum, FDD, Crystal…
- 2001
  - les principaux méthodologues s'accordent sur le « Agile manifesto »
- Depuis
  - projets Agile mixent des éléments des principales méthodes

Du rien au monumental à l'agile

- Rien
  - « code and fix »
  - marche bien sur les petits projets, suicidaire ensuite
- Monumental
  - méthodes, processus, contrats : rationalisation à tous les étages
  - problèmes et échecs
    - trop de choses sont faites qui ne sont pas directement liées au produit logiciel à construire
    - planification trop rigide
- Agile
  - trouver un compromis : le minimum de méthode permettant de mener à bien les projets en restant agile
    - capacité de réponse rapide et souple au changement
    - orientation vers le code plutôt que la documentation

Principes communs des méthodes Agile

- Méthodes adaptatives (vs. prédictives)
  - itérations courtes
  - lien fort avec le client
  - fixer les délai et les coûts, mais pas la portée
- Insistance sur les hommes
  - les programmeurs sont des spécialistes, et pas des unités interchangeables
  - attention à la communication humaine
  - équipes auto-organisées
- Processus auto-adaptatif
  - révision du processus à chaque itération

Méthodes agiles

- Simplicité
- Légèreté
- Orientées participants plutôt que plan
- Nombreuses
  - XP est la plus connue
- Pas de définition unique
- Mais un manifeste

Manifeste Agile

- Février 2001, rencontre et accord sur un manifeste
- Mise en place de la « Agile alliance »
  - objectif : promouvoir les principes et méthodes Agile
  - http://www.agilealliance.com/
- Les signataires privilégient
  - les individus et les interactions davantage que les processus et les outils
  - les logiciels fonctionnels davantage que l'exhaustivité et la documentation
  - la collaboration avec le client davantage que la négociation de contrat
  - la réponse au changement davantage que l'application d'un plan
- 12 principes
  - (transparents suivants)

http://agilemanifesto.org/principles.html

Manifeste Agile : principes

1. Our highest priority is to satisfy the costumer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile process harness change for the customer´s competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

http://agilemanifesto.org/principles.html

Manifeste Agile : principes

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

(Larman 2005, p.36-37)

Processus agile et modélisation

- Utilisation d'UML
- La modélisation vise avant tout à comprendre et à communiquer
- Modéliser pour les parties inhabituelles, difficiles ou délicates de la conception.
- Rester à un niveau de modélisation minimalement suffisant
- Modélisation en groupe
- Outils simples et adaptés aux groupes
- Les développeurs créent les modèles de conception qu'ils développeront

Plan de ce cours

- Principes des méthodes Agile
- **XP : eXtreme Programming**
- Scrum
- Autres méthodes

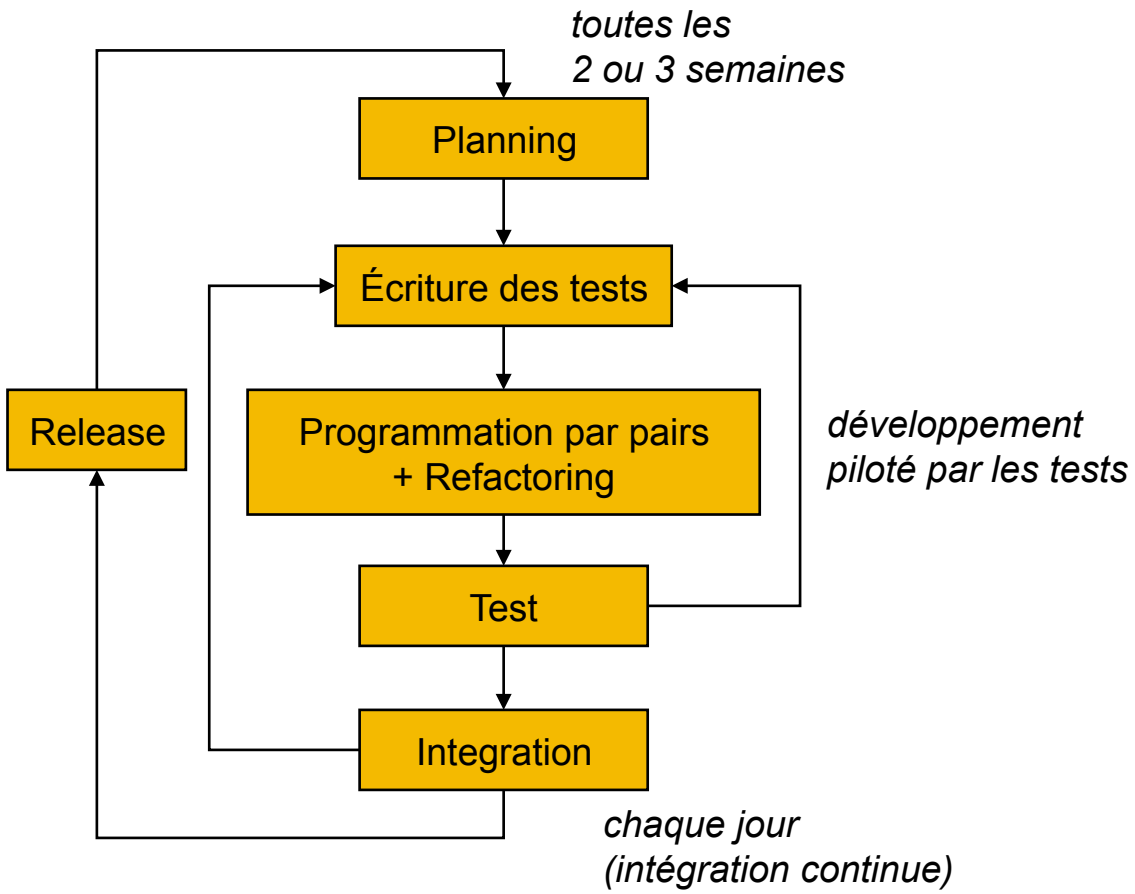
XP (eXtreme Programming)

http://www.extremeprogramming.org/

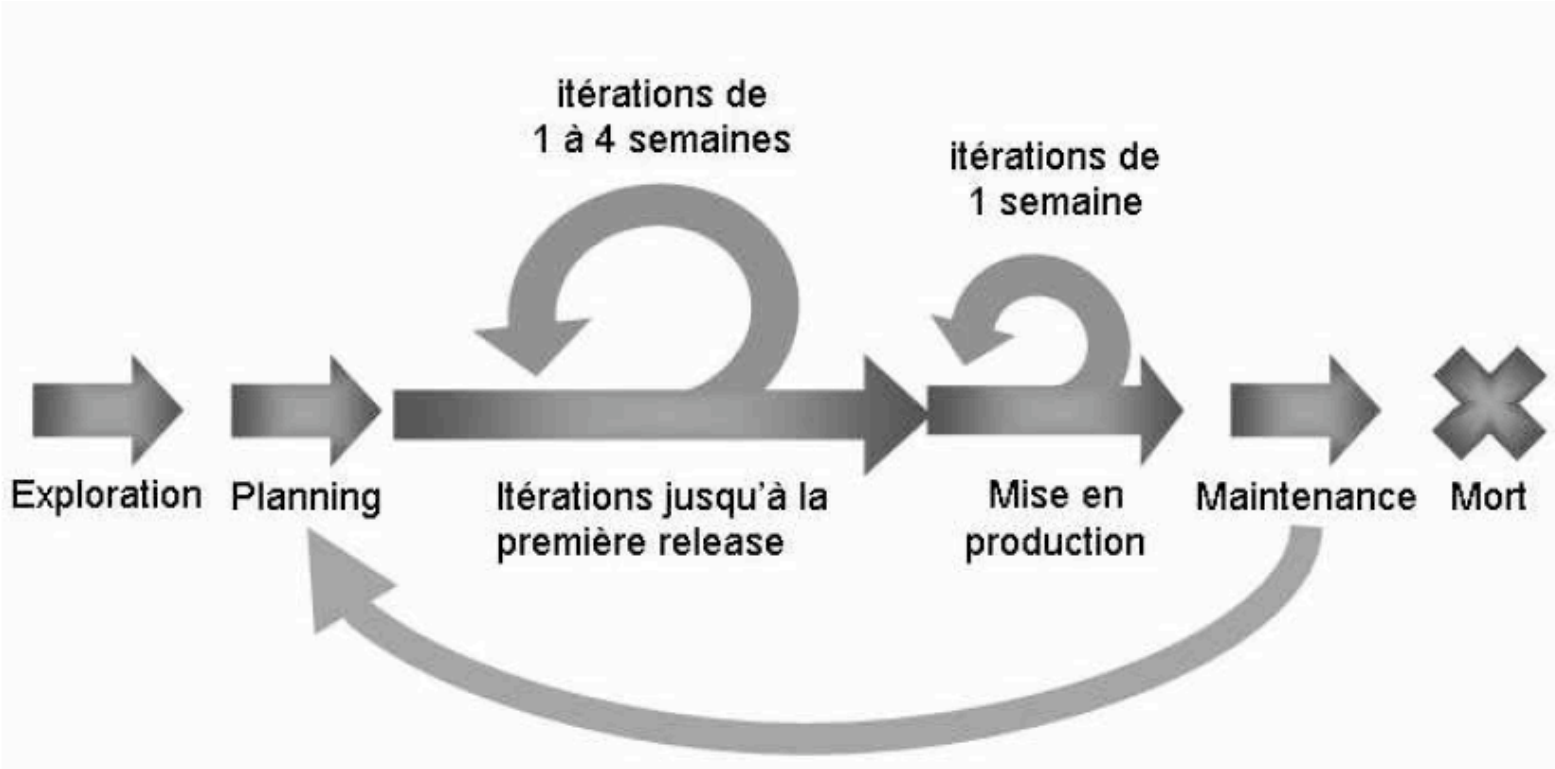
- Caractéristiques principales
  - Le client (maîtrise d'ouvrage) pilote lui-même le projet, et ce de très près grâce à des cycles itératifs extrêmement courts (1 ou 2 semaines).
  - L'équipe autour du projet livre très tôt dans le projet une première version du logiciel, et les livraisons de nouvelles versions s'enchaînent ensuite à un rythme soutenu pour obtenir un feedback maximal sur l'avancement des développements.
  - L'équipe s'organise elle-même pour atteindre ses objectifs, en favorisant une collaboration maximale entre ses membres.
  - L'équipe met en place des tests automatiques pour toutes les fonctionnalités qu'elle développe, ce qui garantit au produit un niveau de robustesse très élevé.
  - Les développeurs améliorent sans cesse la structure interne du logiciel pour que les évolutions y restent faciles et rapides.
  - (extrait de http://www.design-up.com/methodes/XP/)

http://www.swen.uwaterloo.ca/~kostas/ECE355-05/lectures/Lect3-Ch15-Unit2.ppt

XP : vue d'ensemble



Planification globale d projet



http://web.engr.oregonstate.edu/~cook/classes/cs569.agile.ppt

XP : rôles et responsabilités

- Client
  - écrit les histoires et les tests fonctionnels
- Testeur
  - aide le client à écrire les tests, prépare les tests automatiques
- Programmeur
  - écrit les tests et puis code
- Coach
  - aide l'équipe par rapport au processus, expert méthode
- Tracker
  - suit les développement, vérifie que l'équipe ne perd pas la bonne direction
- Manager
  - Responsable infrastructre et soucis extérieurs
- Consultant
  - fournit les connaissances spécialisées au besoin

XP : pratiques (1)

- Le « jeu de la planification »
  - regroupement des intervenants pour planifier l'itération
  - les développeurs évaluent les risques techniques et les efforts prévisibles liés à chaque fonctionnalité (user story, sortes de scénarios abrégés)
  - les clients estiment la valeur (l'urgence) des fonctionnalités, et décident du contenu de la prochaine itération
- Temps court entre les releases
  - au début : le plus petit ensemble de fonctionnalités utiles
  - puis : sorties régulières de prototypes avec fonctionnalités ajoutées
- Métaphore
  - chaque projet a une métaphore pour son organisation, qui fournit des conventions faciles à retenir

XP : pratiques (2)

- Conception simple
  - toujours utiliser la conception la plus simple qui fait ce qu'on veut
    - doit passer les tests
    - assez claire pour décrire les intentions du programmeur
  - pas de généricité spéculative
- Tests
  - développement piloté par les tests : on écrit d'abord les tests, puis on implémente les fonctionnalités
  - les programmeurs s'occupent des tests unitaires
  - les clients s'occupent des tests d'acceptation (fonctionnels)
- Refactoring
  - réécriture, restructuration et simplification permanente du code
  - le code doit toujours être propre

XP : pratiques (3)

- Programmation par paires (pair programming)
  - tout le code de production est écrit par deux programmeurs devant un ordinateur
  - l'un pense à l'implémentation de la méthode courante, l'autre à tout le système
  - les paires échangent les rôles, les participants des paires changent
- Propriété collective du code
  - tout programmeur qui voit une opportunité d'améliorer toute portion de code doit le faire, à n'importe quel moment
- Intégration continue
  - utilisation d'un gestionnaire de versions (e.g., CVS)
  - tous les changements sont intégrés dans le code de base au minimum chaque jour : une construction complète (build) minimum par jour
  - 100% des tests doivent passer avant et après l'intégration

XP : pratiques (4)

- Semaine de 40 heures (35 en France ?)
  - les programmeurs rentrent à la maison à l'heure
  - faire des heures supplémentaire est signe de problème
  - moins d'erreurs de fatigue, meilleure motivation
- Des clients sur place
  - l'équipe de développement a un accès permanent à un vrai client/utilisateur (dans la pièce d'à côté)
- Des standards de codage
  - tout le monde code de la même manière
    - tout le monde suit les règles qui ont été définies
    - il ne devrait pas être possible de savoir qui a écrit quoi

XP : pratiques (5)

- Règles
  - l'équipe décide des règles qu'elle suit, et peut les changer à tout moment
- Espace de travail
  - tout le monde dans la même pièce
    - awareness
  - tableaux au murs
  - matérialisation de la progression du projet
    - par les histoires (user stories) réalisées et à faire
      - papiers qui changent de position, sont réorganisés
    - par les résultats des tests
    - ...

XP : quelques point du site web

- Planning
  - User stories are written.
  - Release planning creates the schedule.
  - Make frequent small releases.
  - The Project Velocity is measured.
  - The project is divided into iterations.
  - Iteration planning starts each iteration.
  - Move people around.
  - A stand-up meeting starts each day.
  - Fix XP when it breaks.
- Designing
  - Simplicity.
  - Choose a system metaphor.
  - Use CRC cards for design sessions.
  - Create spike solutions to reduce risk.
  - No functionality is added early.
  - Refactor whenever and wherever possible.
- Coding
  - The customer is always available.
  - Code must be written to agreed standards.
  - Code the unit test first.
  - All production code is pair programmed.
  - Only one pair integrates code at a time.
  - Integrate often.
  - Use collective code ownership.
  - Leave optimization till last.
  - No overtime.
- Testing
  - All code must have unit tests.
  - All code must pass all unit tests before it can be released.
  - When a bug is found tests are created.
  - Acceptance tests are run often and the score is published.

XP : avantages

- Concept intégré et simples
- Pas trop de management
  - pas de procédures complexes
  - pas de documentation à maintenir
  - communication directe
  - programmation par paires
- Gestion continuelle du risque
- Estimation permanente des efforts à fournir
- Insistance sur les tests : facilite l'évolution et la maintenance

XP : inconvénients

- Approprié pour de petites équipes (pas plus de 10 développeurs), ne passe pas à l'échelle
  - pour des groupes plus gros, il faut plus de structure et de documentation (ceremony)
- Risque d'avoir un code pas assez documenté
  - des programmeur qui n'auraient pas fait partie de l'équipe de développement auront sans doute du mal à reprendre le code
- Pas de design générique
  - pas d'anticipation des développements futurs

Plan de ce cours

- Principes des méthodes Agile
- XP : eXtreme Programming
- Scrum
- Autres méthodes

http://web.engr.oregonstate.edu/~cook/classes/cs569.agile.ppt

Scrum

- Scrum : mêlée
- Phases
  - Initiation / démarrage
    - Planning
      - définir le système : product Backlog = liste de fonctionnalités, ordonnées par ordred de priorité et d'effort
    - Architecture
      - conception de haut-niveau
  - Développement
    - Cycles itératifs (sprints) : 30j
      - amélioration du prototype
  - Clôture
    - Gestion de la fin du projet : livraison…

http://fr.wikipedia.org/wiki/Scrum

Principes Scrum

- Isolement de l'équipe de développement
  - l'équipe est isolée de toute influence extérieure qui pourrait lui nuire. Seules l'information et les tâches reliées au projet lui parviennent : pas d'évolution des besoins dans chaque sprint.
- Développement progressif
  - afin de forcer l'équipe à progresser, elle doit livrer une solution tous les 30 jours. Durant cette période de développement l'équipe se doit de livrer une série de fonctionnalités qui devront être opérationnelles à la fin des 30 jours.
- Pouvoir à l'équipe
  - l'équipe reçoit les pleins pouvoirs pour réaliser les fonctionnalités. C'est elle qui détient la responsabilité de décider comment atteindre ses objectifs. Sa seule contrainte est de livrer une solution qui convienne au client dans un délai de 30 jours.
- Contrôle du travail
  - le travail est contrôlé quotidiennement pour savoir si tout va bien pour les membres de l'équipe et à la fin des 30 jours de développement pour savoir si la solution répond au besoin du client.

http://web.engr.oregonstate.edu/~cook/classes/cs569.agile.ppt

Scrum : rôles responsabilités

- Scrum Master
  - expert de l'applicatron de Scrum
- Product owner
  - responsable officiel du projet
- Scrum Team
  - équipe projet.
- Customer
  - participe aux réunions liées aux fonctionnalités
- Management
  - prend les décisions

http://web.engr.oregonstate.edu/~cook/classes/cs569.agile.ppt

Scrum : pratiques

- Product Backlog
  - état courant des tâches à accomplir
- Effort Estimation
  - permanente, sur les entrées du backlog
- Sprint
  - itération de 30 jours
- Sprint Planning Meeting
  - réunion de décision des objectifs du prochain sprint et de la manière de les implémenter
- Sprint Backlog
  - Product Backlog limité au sprint en cours
- Daily Scrum meeting
  - ce qui a été fait, ce qui reste à faire, les problèmes
- Sprint Review Meeting
  - présentation des résultats du sprint

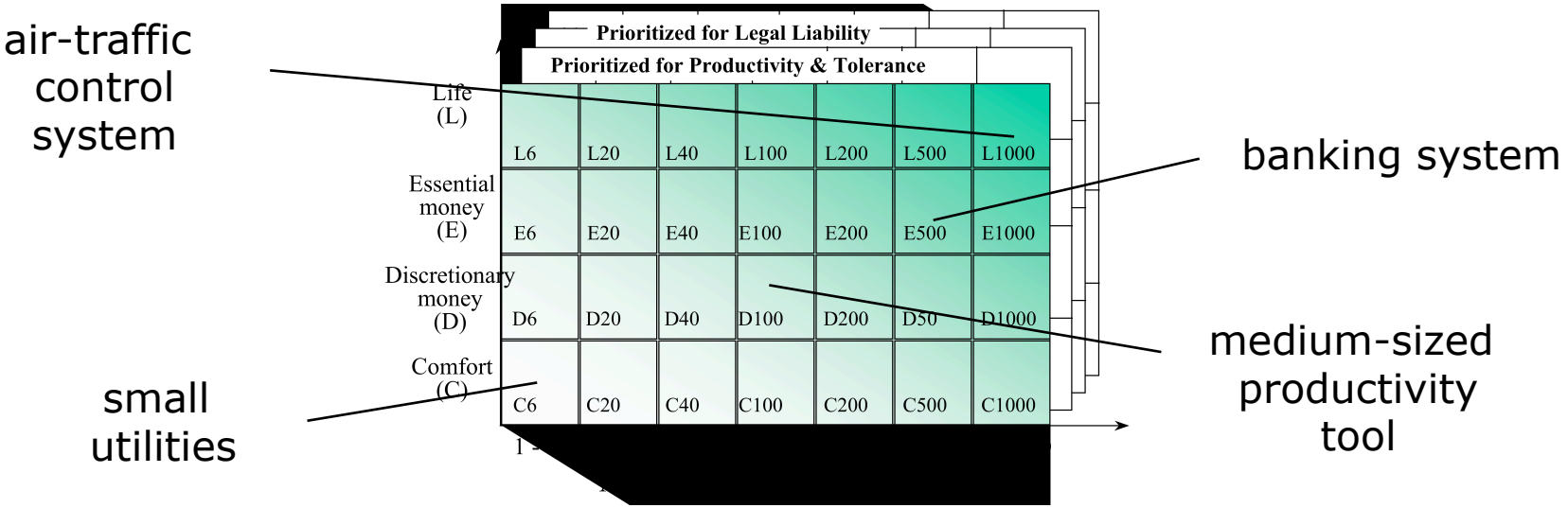
Plan de ce cours

- Principes des méthodes Agile
- XP : eXtreme Programming
- Scrum
- **Autres méthodes**

Méthodologies « Crystal »

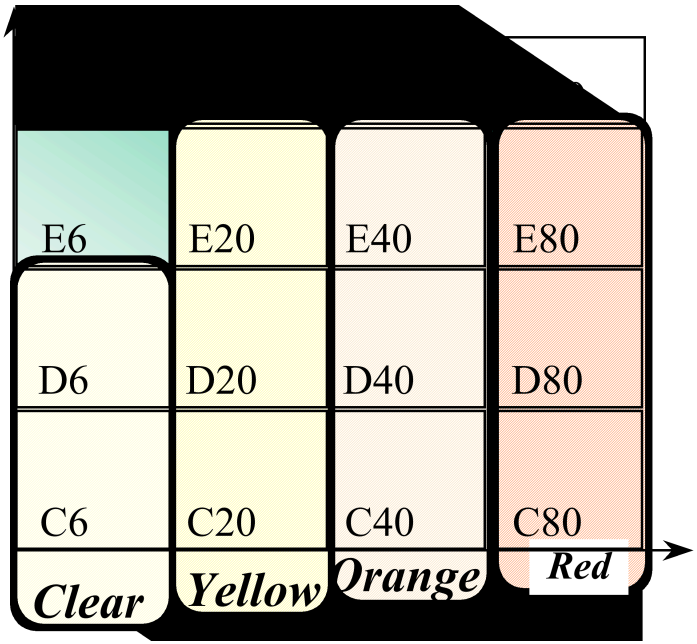
- Alistair Cockburn (2002). Agile Software Development.
  Addison Wesley
- Le développement logiciel vu comme un jeu coopératif de communication et d'invention
  - des projets différents et des méthodes différentes
  - le projet change à mesure que les gens changent
- Choix de la méthode en fonction de différents facteurs
  - taille en nombre de personnes
  - criticité pour le client

Méthodologies « Crystal »



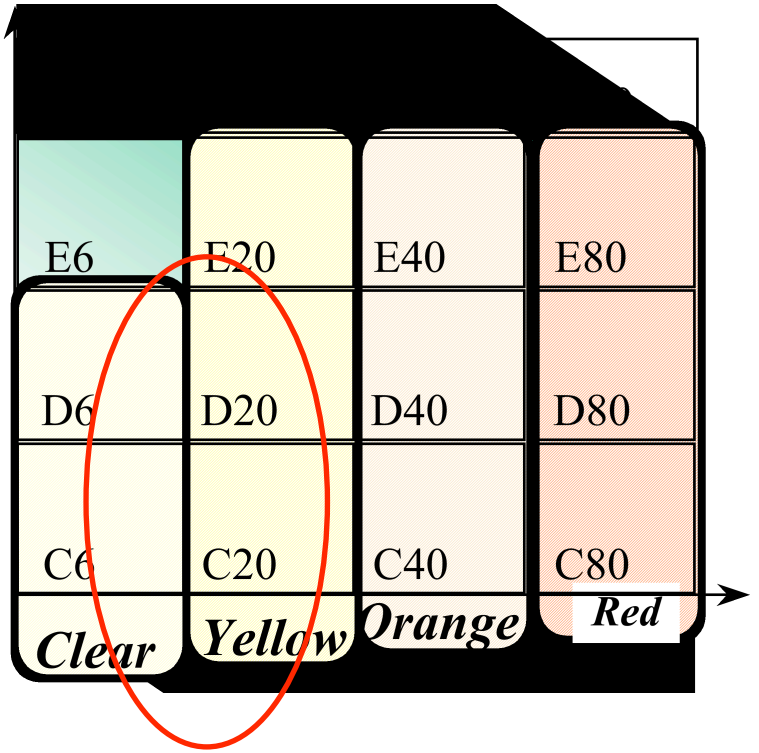
La famille de méthodes Crystal

- Familles conçues à partir de l'observation et des interviews
- Trouver à chaque fois la méthode la moins rigide qui réussira quand même
  - haute productivité, haute tolérance
  - focus sur la communication
- Exemples
  - Crystal Clear
  - Crystal Yellow
  - Crystal Orange
  - Crystal Red



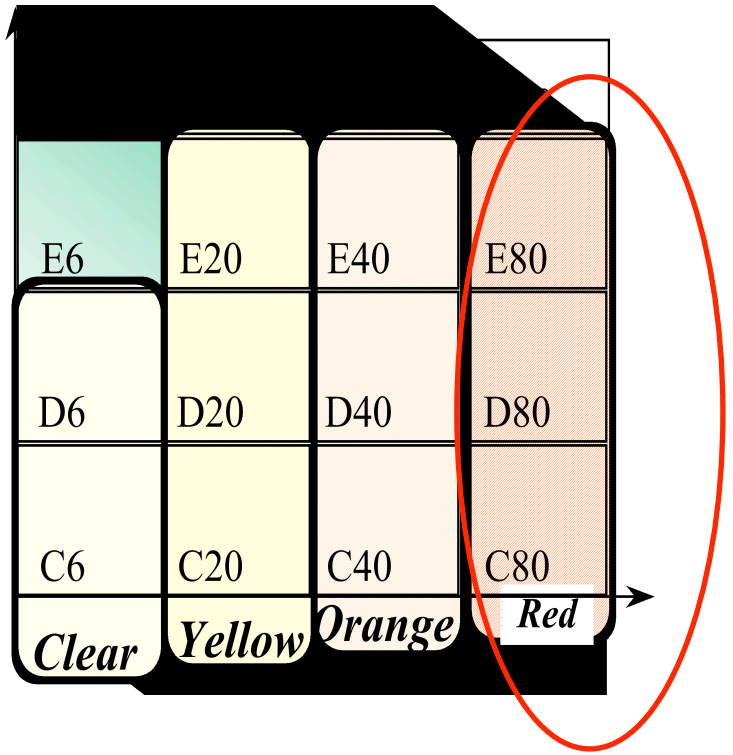
Crystal Clear

- C6-D6"
- Une seule équipe, à un seul endroit
  - *Good, small teams produce good, small software products*
  - Communication étroite
- Rôles
  - sponsor
  - développeur sénior
  - concepteurs-programmeurs
  - utilisateurs
- Livraison incrémentale tous les 2/3 mois



Crystal Orange

- "C40-E40"
  - plus de structure d'équipes
  - lus de coordination
- Tout le monde dans le même immeuble
- Rôles
  - Sponsor
  - Expert métier
  - Expert IHM
  - Expert techniques
  - Analyste concepteur métier
  - Manager
  - Architecte
  - Concepteur mentor
  - ...
- 1-2 ans de développement
- Importance du respect des délais et de l'adaptation au marché



Coders' dojo

- Idée
  - parallèle arts martiaux / conception-programmation OO
    - il faut s'entraîner à appliquer des « routines » connues avant de pouvoir commencer à les utiliser de façon créative, voire à en inventer de nouvelles
    - les débutant doivent apprendre des maîtres
  - un exemple de formation
    - http://www.xpday.net/Xpday2005/CodersDojo.html

Conclusion

- « There is no silver bullet » : le retour

  Même si le développement incrémental permet de s'affranchir de beaucoup de problèmes, il y aura quand même des problèmes. Mais ceux-ci seront normalement d'ampleur plus faible, et mieux gérés.
- Toute méthode est adaptable et doit être adaptée

  Mais, lorsque l'on débute, il vaut mieux ne pas trop s'écarter de la voie décrite pour bien comprendre au départ (cf. musique)

Crédits – remerciements

- J.L. Sourrouille (INSA de Lyon)
- Sources
  - http://www.swen.uwaterloo.ca/~kostas/ECE355-05/lectures/Lect3-Ch15-Unit2.ppt
  - http://web.engr.oregonstate.edu/~cook/classes/cs569.agile.ppt

ANNEXE

NICOLAS CHAN SHIN-YU (ex. étudiant MIAGE)

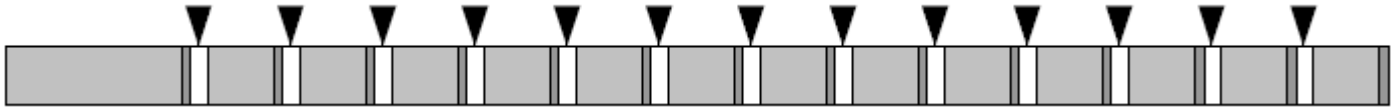
eXtreme Programming

Introduction (1/3)

- Démarche traditionnelle
  - spécification > conception > réalisation > validation
- Garanties au client
- Organisation planifiée du projet
- Problèmes :
  - les besoins du client peuvent changer
  - Erreurs de conception détectées pendant l'implémentation

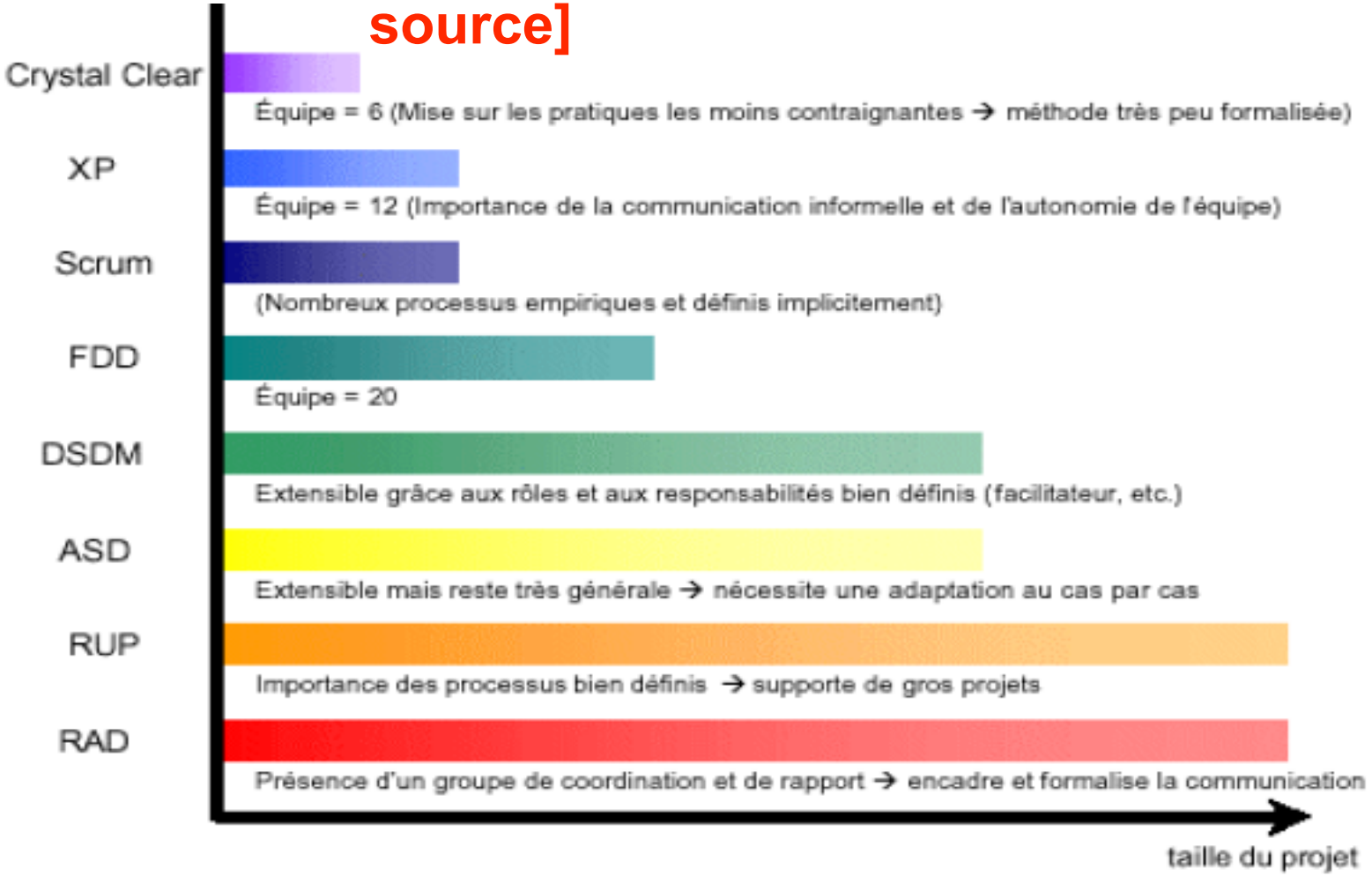
Introduction (2/3)

- Une solution : les méthodes agiles
  - Diffuser le processus de décision tout au long du projet
  - Enchaînement de cycles itératifs très courts



Introduction (3/3)

Comparatif des méthodes [rajouter la source]



Problématique

- Qu'apporte la démarche d'eXtrême Programming ?

Sommaire

- Présentation
  - Principes
  - Structure de l'équipe
- Un projet XP
  - XP sur le plan collaboratif
  - XP et pilotage de projet
  - XP, conception et développement
- Appliquer XP?
- Evolution d'XP

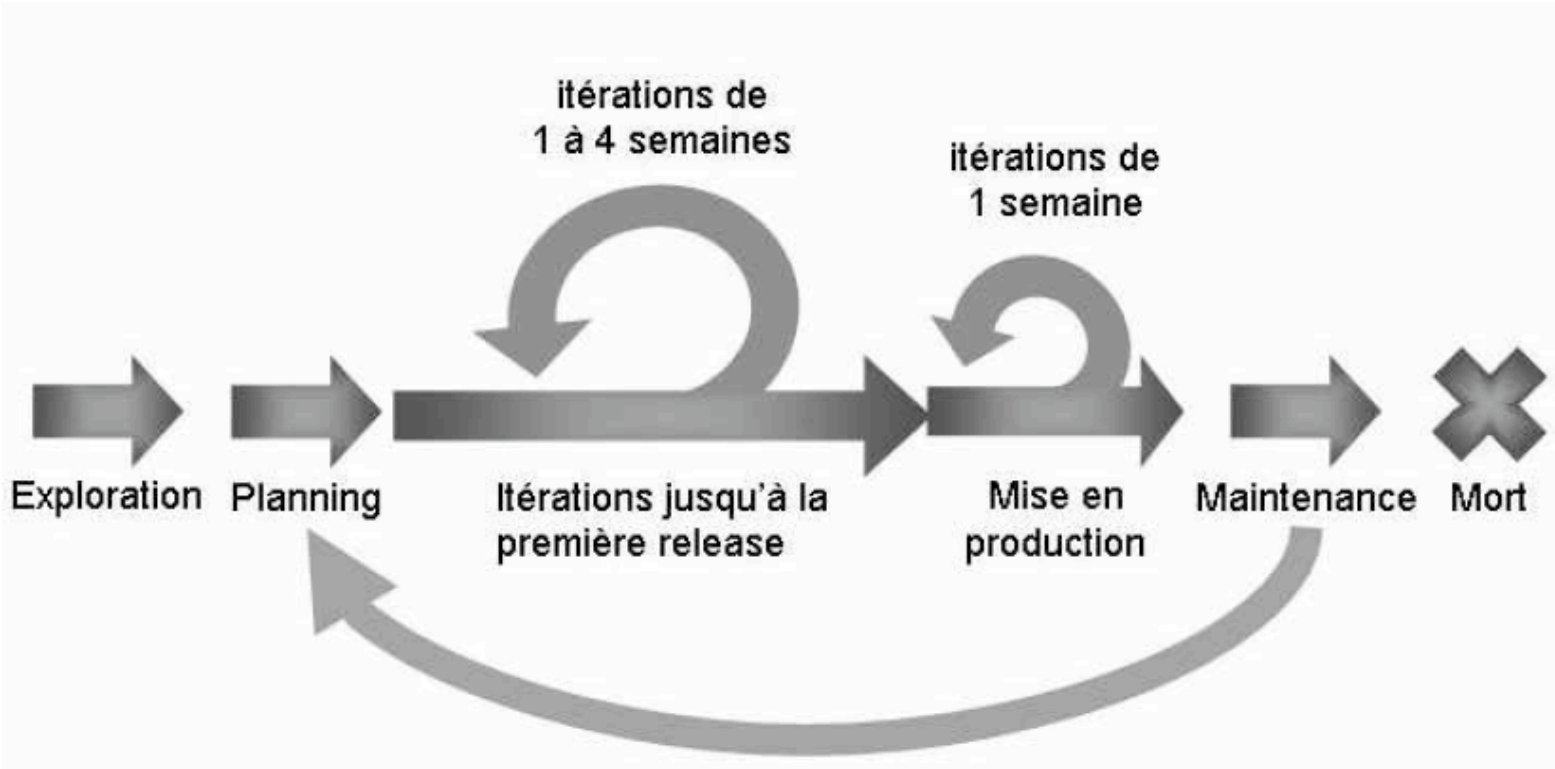
Présentation d'XP

Principes d'XP (1/3)

- Historique
  - 1996 : Ward Cunningham et Kent Beck
  - Projet Chrysler Comprehensive Compensation (C3)
  - Réorganisation du système de paie
  - Propagation mondiale grâce à Internet
  - Implantation lente en France

Principes d'XP (2/3)

- Cycle de vie d'XP **[source]**



Principes d'XP (3/3)

- Facteurs
  - Feedback rapide et constant
  - Compréhension partagée
  - Bien-être de l'équipe
  - Processus fluide et continu

Aspect humain & collaboratif

collaboration

Les principes propres à **12 principes** :

1. User de la métaphore
2. Intégration continue
3. Propriété collective du code
4. Convention du codage
5. Programmation en binôme

Aspects humains & collaboratif

XP et son aspect humain

- Dimension humaine = souvent la plus déterminante pour la réussite de tout projet

- **Aspects importants:**
  - Le comité de pilotage
  - Les différents profils
  - Le rôle de chaque profil
  - L'unification des savoirs : « la formation »
  - Les règles de collaboration

Le comité de pilotage (2/2)

- **Comité de pilotage :**
  - dirige les aspects stratégiques du projet,
  - définit les objectifs
  - et alloue les moyens nécessaires

- **Client interlocuteur :**
  - le plus à même de connaître le besoin et de l'exprimer
  - « Client interlocuteur » = appui du « sponsor principal »
    - directeur des systèmes d'information
    - ou porte-parole officiel de l'entreprise disposant d'une autorité suffisante pour légitimer le projet et incarner l'engagement de l'entreprise
- **Phase de préparation, il faut s'assurer de:**
  - la clarté des objectifs du projet
  - Leur compréhension par toutes les parties prenantes
  - Accord de l'entreprise cliente

Le comité de pilotage (2)

- **Assurer le bien-être de l'équipe**

XP est une méthode itérative qui met en place une mécanique de production de logiciel fonctionnant de manière régulière et continue.

- **Une pratique humaniste en favorisant :**
  - Une élimination progressive des risques
  - Établir un rythme de travail régulier, sans heures supplémentaires

- **But : Éliminer**
  - Le stress,
  - Tensions a l'intérieur du groupe,
  - Tensions en le groupe et les donneurs d'ordres,
  - Éliminer le risque de défection.

Structure de l'équipe

Profils intervenants dans une équipe XP :

- Le client
- Le testeur
- Le manager
- Le coach
- Le tracker
- Le programmeur

Le rôle de chaque profil (1/2)

- Le client :
  - Membre à part entière de l'équipe
  - Présence physique imposé dans l'équipe tout au long du développement
  - Spécifie les fonctionnalités à implémenter et tests fonctionnels
  - Rôle pouvant être tenu par une ou plusieurs personnes
- Le testeur :
  - Assistant du client, = un programmeur
  - Implémente (code) les tests de recette le plus tôt possible, valide le fonctionnement du code et vérifie la non régression
- Le manager :
  - Responsable de l'infrastructure dans laquelle l'équipe travaille
  - S'assure la non existence de problèmes étrangers au projet ou logistiques (espace de travail, outillage, documentation, Etc.)

Le rôle de chaque profil (2/2)

- Le coach :
  - S' assurer la bonne compréhension de la méthode XP et son application correcte par les différents acteurs du projet
  - Généralement « expert méthode » et bon communicateur doublé d'un technicien crédible et respecté
- Le tracker :
  - Contrôle l'avancement des taches à l'intérieur d'une itération.
  - S'entretient fréquemment avec chaque programmeur pour s'enquérir des difficultés rencontrées et du travail déjà effectué.
  - Construire régulièrement une vision fiable de l'avancement des taches afin de détecter le plus tôt possible les dérives et de lancer une nouvelle phase si nécessaire
- Le programmeur :
  - Estime la charge nécessaire à l'implémentation d'un scénario dans le cadre du jeu de la planification.
  - Il implémente les scénarios en s'appuyant sur l'écriture de tests unitaires
  - Il est aussi analyste, concepteur.

L'unification des savoirs

Formation

- **Mise à niveau des ressources avant le lancement du projet :**
  - Phase d'apprentissage théorique par un formateur professionnel
  - Mise en pratique dans le cadre du projet pilote
- **Projet partagé en 2 phases :**
  - **Coaching,** peu productive. Objectif :
    - réaliser en situation le potentiel de chaque membre
    - Gérer les parcours individualisé avec le soutien pédagogique de moniteurs (les coachs)
  - **Fonctionnement productif** du mode coaching lors de la première phase du vrai projet

La montée en compétence

- **Maximiser la rentabilité de l'investissement** pour la montée en compétences des ressources
- **Éviter l'écueil 80/20** : l'acquisition des 20% de connaissances les plus pointues requiert 80% des efforts (et de l'investissement) en formation.
  → Faire confiance au temps et considérer pour l'uniformisation des compétences collectives

Les règles de collaboration(1/2)

Exigence de qualité du code produit

- Conventions de codage :
  - Homogénéité du niveau et style de codage des développeurs
  - Utilisation d'outils de mise en forme automatique du code basés sur des conventions codage paramétrables
  - Intégration continue

Les règles de collaboration (2/2)

- Programmation en binôme des développeurs

  But :

  - se contrôler mutuellement
  - diminuer les erreurs de conception
  - Meilleure concentration sur le travail
  - Éliminer le risque de dépendance à un développeur
  - Lisser les inégalités d'expériences en associant « confirmé & débutant »

→ Favorise la circulation de la connaissance à l'intérieur du projet.

Pratiques de type pilotage de projet

Le jeu de planification (1/3)
(Planning Game)

- Choix d'un rythme de livraison (2-3 mois)
- Choix du rythme des itérations (1 à 3 semaines)
- Étapes :

1. **Le client décrit ses besoins: *user stories***
   Histoires simples, dans le langage du client, décrivant les fonctionnalités demandées. Un projet peut comporter une centaine de user stories.

2. **Les développeurs estiment le coût d'implémentation**
   En équipe avec la collaboration du client. On note en "semaines idéales". Une histoire trop grande (>3 points) doit être partagée. Si on ne sait pas estimer, on explore le sujet en faisant un essai ("*spike*").

Le jeu de planification (2/3)
(Planning Game)

3. Le client choisit les histoires à implémenter

- Le client trie les histoires par priorité.
- Les développeurs déclarent leur *vélocité* (performance réelle mesurée à chaque itération).
- Calcul du nombre de points que l'équipe peut traiter en une itération.
- Le client choisit les histoires à implémenter dans la prochaine livraison.

4. Les développeurs implémentent les histoires

Planning d'une itération:

→ pour chaque histoire prévue dans la livraison, les développeurs déterminent les tâches concrètes et les estiment en "jours idéaux".

→Les développeurs acceptent les tâches et les implémentent.

Le jeu de planification (3/3)
(Planning Game)

- **Avantages**
  - Confrontation et homogénéisation des connaissances métier par les développeurs.
  - La priorisation des scénarios permet d'avoir rapidement un noyau fonctionnel

- **Limites**

Il est possible qu'une inconnue technique risque de limiter la fiabilité de l'évaluation de la charge donnée par les développeurs (lors de la phase d'exploration)

Livraisons fréquentes (1/2)
(Frequent Releases)

- **Principe**

  Livrer en fin de chaque itération, le rythme des livraisons en est d'une toutes les quelques semaines

- **Avantages**

  - Donner au client le plus fréquemment possible, un aperçu sur l'état d'avancement de l'application
  - La possibilité de corriger rapidement d'éventuelles erreur ou mauvaises interprétations
  - Diminuer la quantité de travail à valider

Livraisons fréquentes (2/2)
(Frequent releases)

- **Limites**

  - Techniquement, les livraisons fréquentes demandent d'avoir un mécanisme d'intégration fiable:
    - Performant (compatible avec la fréquence de livraison)
    - Automatisé (reproduire une livraison antérieure)

  - Fréquence VS précipitation

  ➔ Sacrifier certaines étapes pour tenir les délais (intégration, tests, ..)

Client sur site (1/2)

- **Avantages**
  - Fiabiliser la communication des besoins aux développeurs et accélérer le processus d'une manière générale.
  - Élimination de la charge de production des documents de spécifications.
  - Le client bénéficie d'une meilleure visibilité sur l'avancement du projet.
  - Tout problème qui pourrait nécessiter une remise en question des priorités (lot de scénarios à implémenter) peut être traité sans délai.

Client sur site (2/2)

- **Limites**
  - C'est l'une des exigences de XP les plus complexes à réaliser dans la pratique
  - En pratique, il arrive que le client présent sur le site du projet ne dispose pas de:

    - La connaissance pour répondre aux questions des développeurs
    - Le pouvoir et l'autorité nécessaires à la prise de décision

  ➔ la fluidité du processus est remise en cause
  ➔ l'avantage de sa proximité s'en trouve considérablement réduit

  - Avoir une personne consacrée au rôle de client XP est un luxe

Assurer le bien-être de l'équipe (1/2)

- **Principes**

- XP s'attache à établir un rythme de travail régulier, sans heures supplémentaires.
- XP est une méthode itérative, tend à mettre en place une mécanique de production fonctionnant de manière régulière et continue:
  - ➔ favoriser une élimination progressive des risques
  - ➔ éviter les « coups de *bourre* »
- XP reconnaît que toute surcharge de travail a un coût:
  - moral (si le stress a un impact sur la motivation de l'équipe)
  - diminution de productivité (qui suit toujours une phase de forte activité)

Assurer le bien-être de l'équipe (2/2)

- **Avantages**

  - Impact positif sur les performances de l'équipe
  - Elle vise à éliminer le stress, les tensions :
    - à l'intérieur du groupe
    - entre le groupe et les donneurs d'ordres
  - Elle contribue ainsi à diminuer les risques de défections

- **Limites**

  Cette pratique, peut-elle vraiment être mis en place?

XP au niveau technique

Architecture

- XP propose une démarche basée sur l'enchaînement suivant:
  - Ecriture d'un test
  - Conception, **un peu**
  - Programmation, un peu
  - Refactoring, un peu.
- XP peut utiliser une technique d'analyse ou de conception, telle que la modélisation avec UML
- XP a une préférence pour la technique des CRC Cards (Class Collaborators Responsabilities)
  - permet plus de communication dans l'équipe
  - architecture est présentée sous forme de métaphore
  - métaphore a le mérite d'être plus synthétique et parlante

Règles de programmation

- Une classe appartient à une seule et unique personne
- Mais le code de l'application appartient à toute l'équipe (consultable par tous)
- Développement en binôme
- Le code créé doit être composé de petites classes et de méthodes courtes
- Programmation pilotée par les tests
- Aucune duplication de code ne doit exister
- Présence obligatoire de commentaires dans le code

➔ **Obtention d'un code homogène pour toute l'application**

Nommage et langage

- Respecter des normes de nommage pour les variables, méthodes, objets, classes, fichiers
- Langage idéal :
  - orienté objet (C++ et le JAVA)
  - des cycles de développement courts
  - permettant la mise en œuvre rapide de tests

XP sur le plan technique

- **Avantages**

- De nombreux tests tout au long du process (unitaires & recettes)

- Interaction importante avec le client

XP sur le plan technique

- **Inconvénients**

- Tests unitaires peuvent prendre le dessus sur la programmation de l'application

- Programmation itérative ne donne pas une vision d'ensemble

Choisir XP?

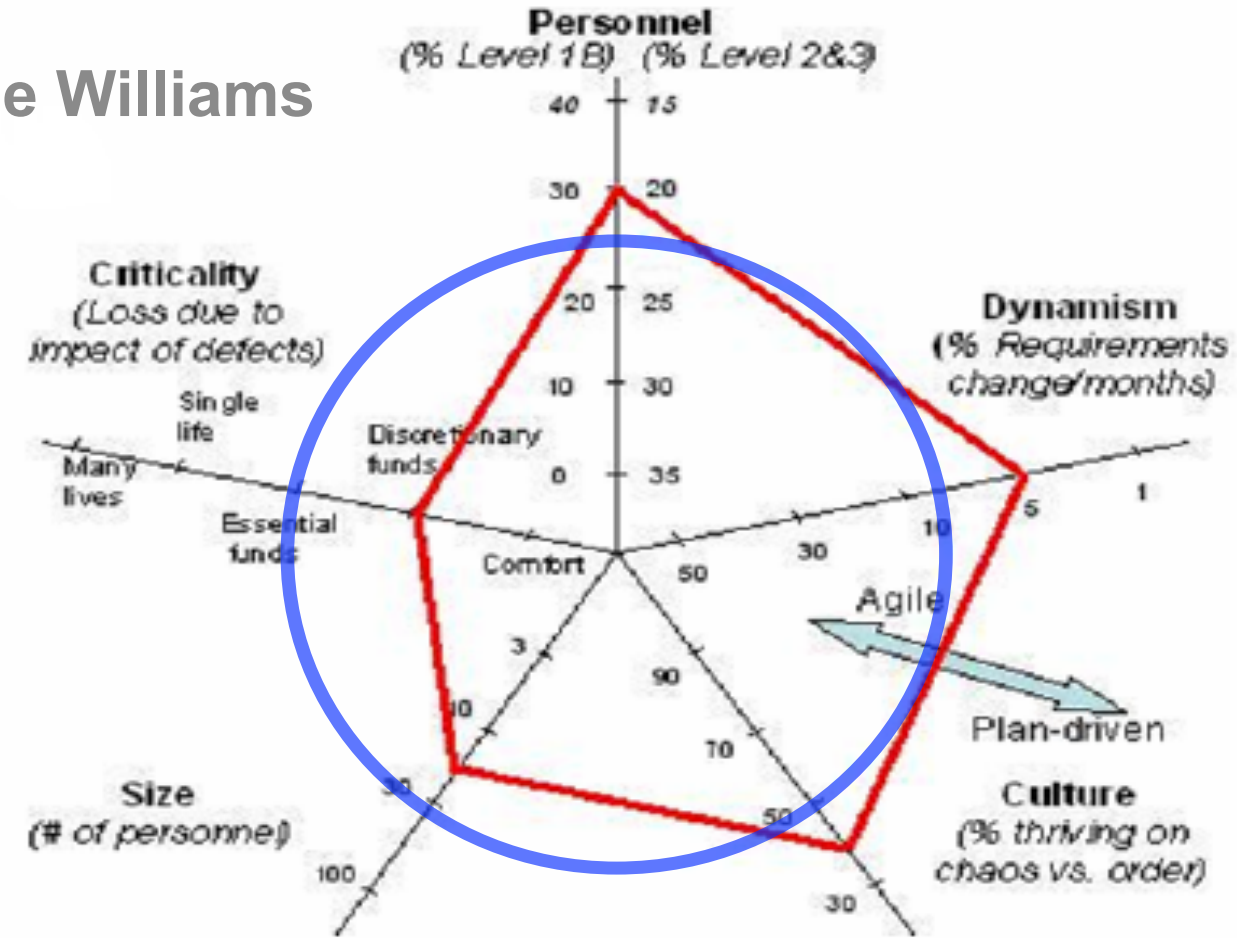
Les paramètres à prendre en compte

Choisir XP?
Les 5 axes

- Expérience
  - Les équipes avec plus d'expérience doivent-ils faire de l'XP?
- Dynamisme
  - Les requirements sont-ils « volatiles »?
- Culture
  - Les membres préfèrent-ils le travail « chaotique »?
- Taille
  - Peut-on faire de l'Agile avec une équipe de cette taille?
- Criticité

Choisir XP?
Exemple: Sabre Airline Solutions

[© Laurie Williams 2006]



Évolution de XP

- Méthodologie ***lightweight***
  - Un blocage culturel
- Une méthode peu applicable
  - Développeur: compétence suffisante
  - Équipe: communication & collaboration efficace
- Une très forte implication
  - Responsabilité du coach
  - Le client XP

Conclusion

- Encore à une phase de lancement
- Une nouvelle vague de méthodes:
  - Pragmatisme
  - Orientation humaine

Bibliographie

- Livres :
  - Pierre-Yves Cloux , RUP, XP : Architecture et outil, industrialiser le processus de développement, DUNOD 2003
  - Layman, L., Williams, L., Cunningham, L., Motivations and Measurements in an Agile Case Study, Journal of System Architecture, to appear.
- Site internet :
  - http://extremeprogramming.free.fr
  - http://www.univ-angers.fr/docs/etudquassi/Methodes_agiles.pdf
  - http://www.commentcamarche.net/genie-logiciel/methodes-agiles.php3
  - http://www.technicmania.com/
  - http://www.extremeprogramming.org/rules/iterative.html
  - http://www.club-java.com/Public/slides/xp/methode**agiles**-clubjava.pdf
  - http://www.isr.uci.edu/events/dist-speakers05-06/williams06.pdf