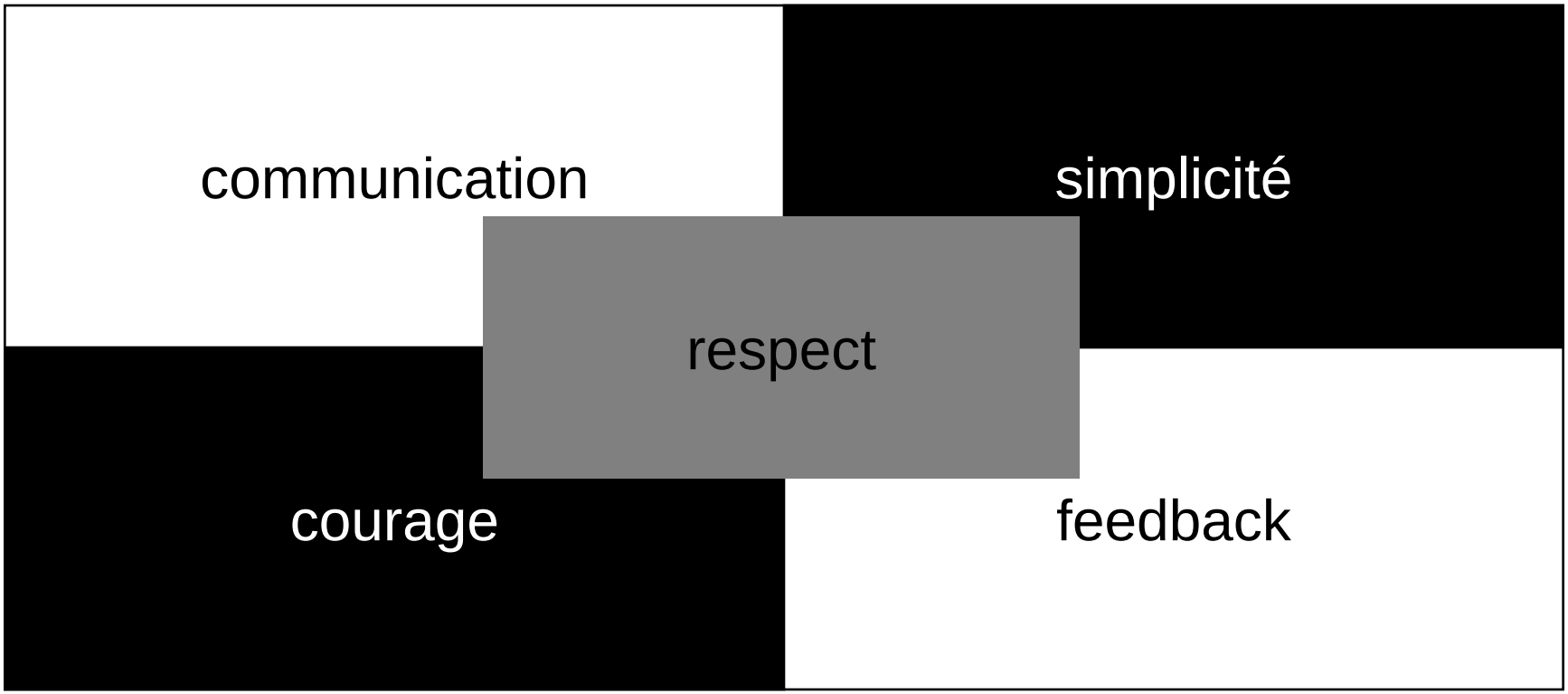
Introduction à
eXtreme Programming

Extreme Programming (XP)

- **L'autre méthodologie dont on parle beaucoup**
- **Comme la plupart des autres méthodes XP est recommandé pour…**
  - Des équipes de tailles petites ou moyennes
  - Des projets avec de nombreuses zones d'incertitude (risques)
  - Lorsque les besoins sont vagues ou évoluent rapidement
- **Adopte des processus issus d'autres méthodologies**
  - E.g. SCRUM, Crystal
- **A été utilisé ou expérimenté chez…**
  - Daimler-Chrysler, Ford, Capital One, IBM, et beaucoup d'autres

Les valeurs

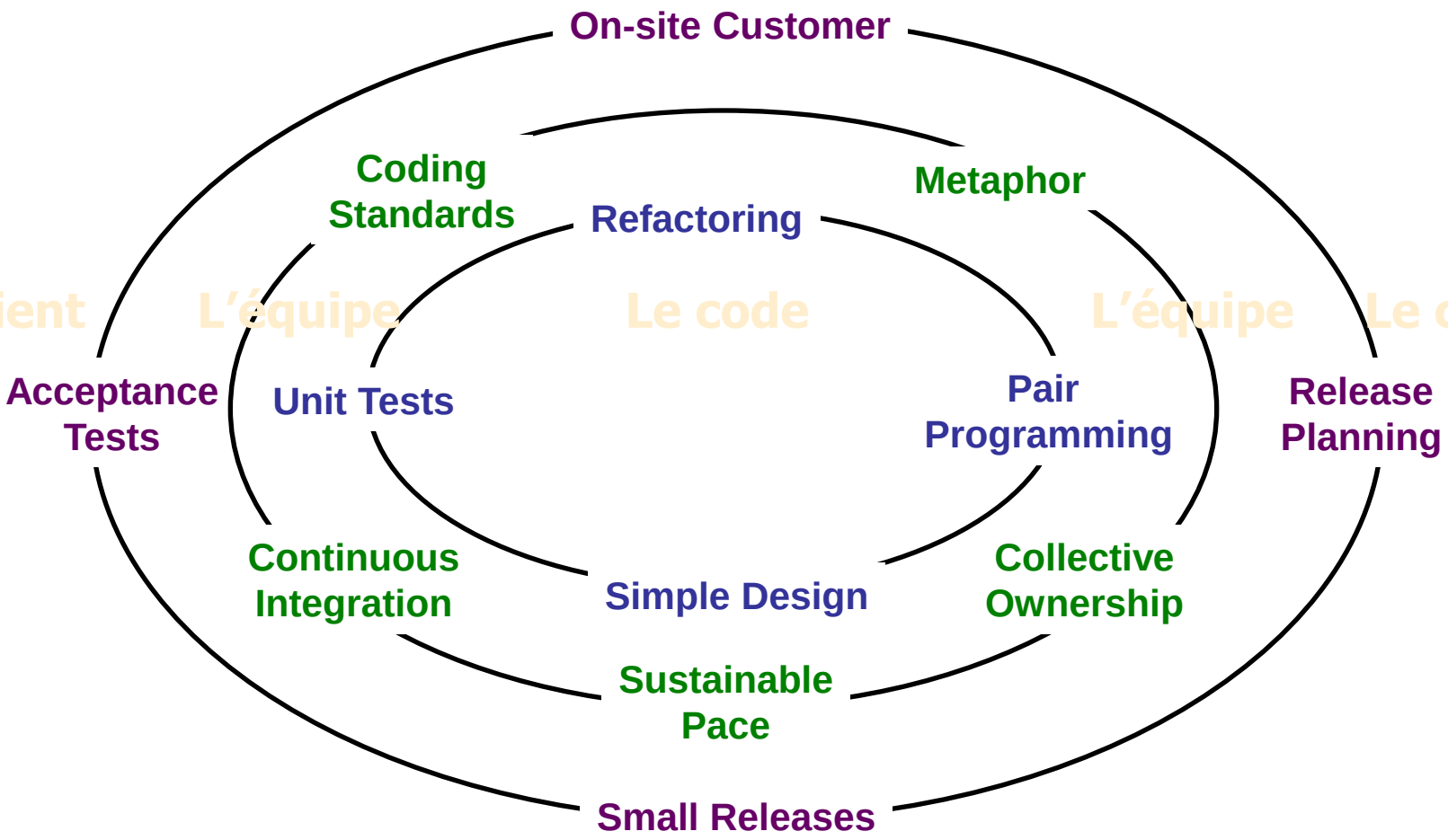


Communication leads to valuable feedback which encourages simplicity which allows for courage to change

Les pratiques

- **Les pratiques découlent des 4 valeurs**
  - Communication, Simplicité, Feedback, Courage
  - Considérez les comme des « des fonctions de maximisation »
- **Pratique = Étude**
  - En musique classique, une étude est, à l'origine, une pièce destinée à améliorer certains aspects techniques d'un élève ou d'un interprète
  - Il arrive de ne pas appliquer toutes les pratiques tout le temps en fonction des projets
  - Mais l'utilisation répétée des pratiques XP améliore incontestablement les compétences des développeurs et leur capacité à proposer des solutions plus rapidement
- **Les pratiques procurent les meilleurs résultats lorsqu'elles sont utilisées de façon conjuguée (SYNERGIE)**

Les « cercles de vie »



Client sur site

- **Définit les besoins, les fonctionnalités, définit les priorités et de répond aux questions des développeurs**
- **Une interaction quotidienne de vis-à-vis**
  - Réduit la quantité de documentation écrite
  - Et le coût élevé de sa création et de sa maintenance

Planification de Release

- **Où « jeu du planning »**
- **Le « client » XP de doit définir la valeur métier des fonctionnalités**
  - User Stories
- **Les développeurs (pas seulement un CP) estiment les fonctionnalités**
- **A partir de ces informations, le client et les développeurs effectuent une sélection en fonction du ration coût / bénéfice**
  - Permet une priorisation optimale des fonctionnalités

Livraisons courtes

- **Mettre en production un système simple le plus rapidement possible puis appliquer des cycles de livraisons très courts**
- **Exemple**
  - Releaser tous les 2 ou 3 mois
  - Chaque release contient plusieurs itérations
- **Établir un « rythme »**
  - Feedback régulier pour le client et l'équipe
- **Permet d'évaluer la véritable valeur du produit dans son environnement réel**

- **Où « Test First », « Test Infected »**
  - TESTS D'ACCEPTANCE : on demande aux clients de fournir des tests d'acceptance avant les développements (idéalement automatisés)
  - TESTS UNITAIRES : les développeurs écrivent d'abord des tests puis créent le logiciel qui répond aux exigences capturées dans les tests
    - Automatisation, Automatisation, Automatisation, (JUnit, XUnit) [5]
  - Un développement piloté par les exigences garantit que les fonctionnalités essentielles seront fournies

Refactoring

- **Les développeurs restructurent le système sans en modifier le comportement pour supprimer les duplications, faciliter la communication, simplifier ou améliorer la flexibilité**
- **Petites étapes**
- **Coder, tester, refactorer, tester, coder, refactorer**
  - Beck suggère [1] des cycles de (10 minutes)
- **Un alignement sur un pattern de conception est un refactoring typique**
- **Basé sur le travail de Martin Fowler [6,7]**

Programmation en binôme

- **Le code est écrit par 2 développeurs sur 1 seule machine**
  - L'un la tactique, l'autre la stratégie
- **Le binôme devrait être « dynamique »**
  - Les membres en binôme changent de rôle toutes les 30 à 60 minutes
  - Et sur tous les types de tâches
- **L'expérimentation montre une réelle efficacité [8]**

Programmation en binôme

- **Les bénéfices**
  - Revue de code continue
  - Partage continu du métier
  - Amélioration continue des compétences techniques (Java, Design Patterns, Refactoring, IDE…)
- **Les développeurs on parfois plus de mal avec cette pratique que les managers**
  - Tester pour voir si cela fonctionne
  - Peut nécessiter un temps d'acclimatation
  - Expérience du développement plus intense

Conception Simple

- **Basé sur le principe que la plus haute valeur métier dérive du programme répondant aux besoins courants le plus simple**
- **Pas d'over-engineering !**
- **K.I.S.S (un concept difficile à appliquer)**
- **2 philosophies communes d'équipes XP**
  - DTSTTCPW – Do The Simpliest Thing That Could Possibly Work
  - YAGNI – You Aren't Gonna Need IT

Standards de développement

- Les développeurs produisent du code en conformité avec des règles favorisant la communication au niveau du code
- L'objectif : produire un code auto documenté
- La raison : le langage comment est le code
- Plus que de la Javadoc; de la bonne Javadoc et des commentaires appropriés

Métaphore

- La vision de « l'architecture » selon XP
- Guide les développements en fournissant une vision commune et unique de la façon dont le système fonctionne
- Définit un vocabulaire et guide l'équipe dans les développements et la communication

Intégration Continue

- Intégrer et construire le système aussi souvent que possible
- Intégrer à chaque fois qu'une tâche est finie
- Permet de connaître à tout instant le « statut » du système

Propriété collective

- **Tout développeur peut modifier toute portion de code à n'importe quel moment**
- **Ceci est facilité par l'utilisation de Standards de développement, de TDD et de Pair Programming (Synergie)**
- **Sécurise l'équipe en terme de vacances, congés, maladie, turn-over**
  - Les progrès ne s'arrêtent pas sur un composant parce qu'un membre de l'équipe est absent

Rythme soutenable

- **Des développeurs fatigués produisent la plupart du temps un code de moindre qualité**
- **Minimiser les « heures supplémentaires » et conserver une équipe fraîche produit un code de meilleur qualité en moins de temps**
- **40 – 45 devrait être la règle**
  - Selon les préférences des équipes
- **Corollaire : ne jamais faire des heures supplémentaires une seconde semaine d'affilée**
  - Éviter le burnout

Avantages d'XP

- **Extrême = continuellement**
  - Revue de code (pair programming)
  - Tests unitaires
  - Tests d'intégration
  - Tests d'acceptance utilisateurs
- **Pouvez vous satisfaire votre client avec du feedback, de la communication et de la simplicité constante ?**
- **Le feedback procuré par les estimations, les tests, les livraisons fréquentes**
  - Donne confiance aux utilisateurs
  - Donne confiance à l'équipe
  - Donne confiance au management

Limitations

- **XP peut ne pas fonctionner pour**
  - Des grosses équipes
  - Des équipes distribuées
  - Des systèmes trop complexes
  - Des projets d'intégration avec du code existant
  - Des organisations où la méthodologie est particulièrement rigide
  - Des organisation ou règne la culture du burn-out
    - Où la valeur d'une personne dépend du temps passé au travail
  - Des ego sur dimensionnés
  - Des environnements physiques inappropriés
- **Mais XP peut être adapté**

Inconvénients d'XP

- **Le changement culturel peut être difficile pour les développeurs et les managers**
  - Mineurs : rythme soutenable, standards de développement, tests, refactoring
  - Majeurs : client sur site, jeu de planification, livraisons fréquentes, conception simple
  - Extrêmes : propriété collective, programmation en binôme, métaphore
- **Requiert des développeurs expérimentés**
- **Disposer d'un client sur site n'est pas toujours possible**
  - Manque de disponibilité
  - Niveau de prise de décision
  - En rupture avec la culture interne

Le cycle standard d'XP

1. Le client écrit ses besoins sous forme de scénarios.

2. Les développeurs évaluent le coût de chaque scénario, en collaboration avec le client.

3. Le client choisit les scénarios à intégrer à la prochaine livraison.

4. Chaque développeur prend la responsabilité d'une tâche pour la réalisation d'un scénario.

5. Le développeur choisit un partenaire.

6. Le binôme écrit les tests unitaires correspondant au scénario à implémenter.

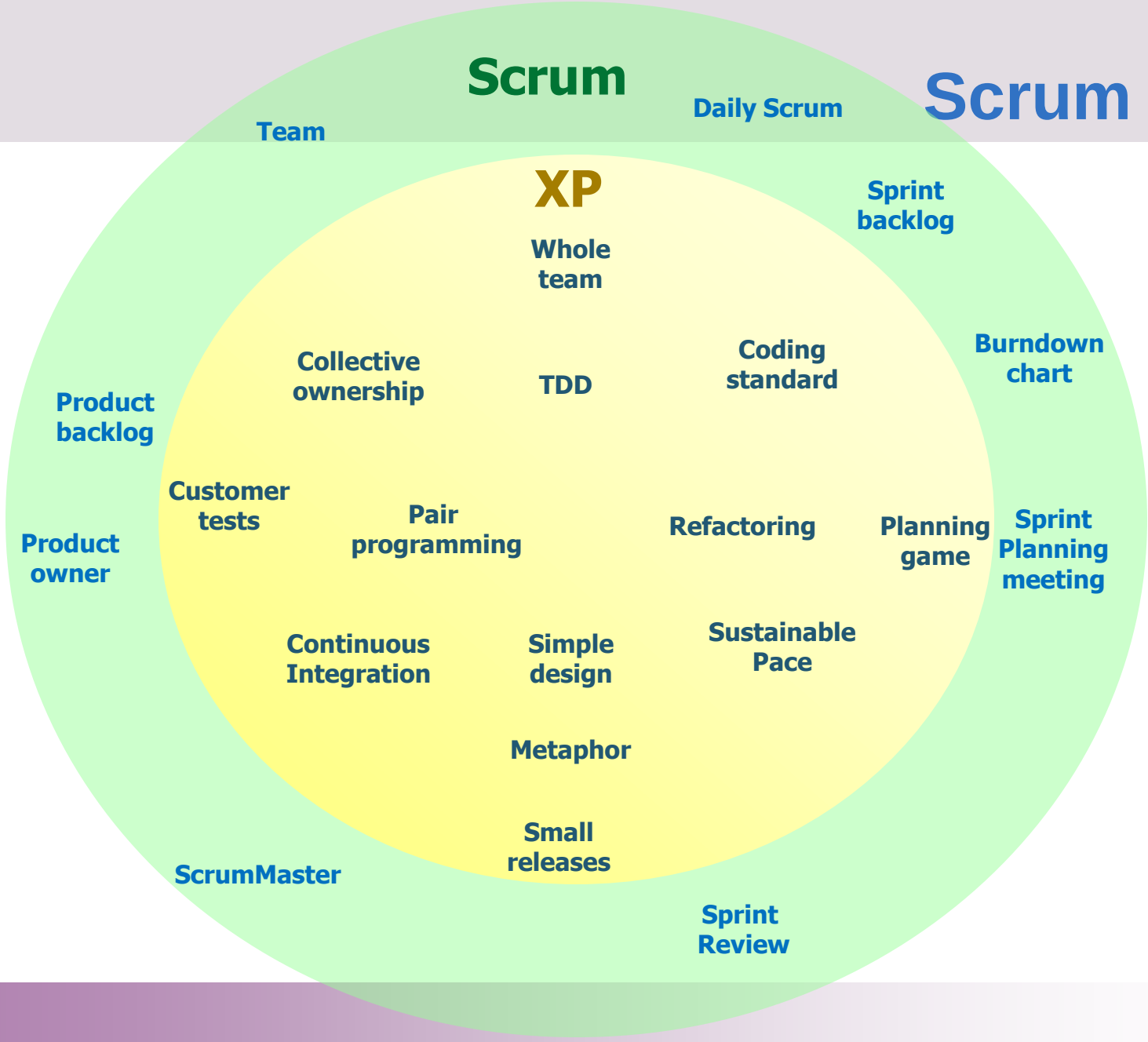
7. Le binôme prépare l'implémentation en réorganisant le code existant, puis il procède à l'implémentation proprement dite.

8. Le binôme intègre ses développements à la version d'intégration.

Planification collective

- Le projet est décomposée en une suite de « tours » appelés **itérations** (durée de 1 à 3 semaines selon les projets) et de **livraisons** au rythme d'une livraison toutes les 2 ou 3 itérations en général.
- Le **cycle des livraisons** porte sur la planification des fonctionnalités décrites sous forme de scénarios client (*user stories*).
- Le **cycle des itérations** porte sur la planification des activités (tâches) menées par les développeurs.
- Les **tâches** sont donc les activités des développeurs qui permettent de réaliser un scénario client.

Scrum & XP


Scrum
Daily Scrum
Team
XP
Sprint backlog
Whole team
Burndown chart
Collective ownership
TDD
Coding standard
Product backlog
Customer tests
Pair programming
Refactoring
Planning game
Sprint Planning meeting
Product owner
Continuous Integration
Simple design
Sustainable Pace
Metaphor
Small releases
ScrumMaster
Sprint Review