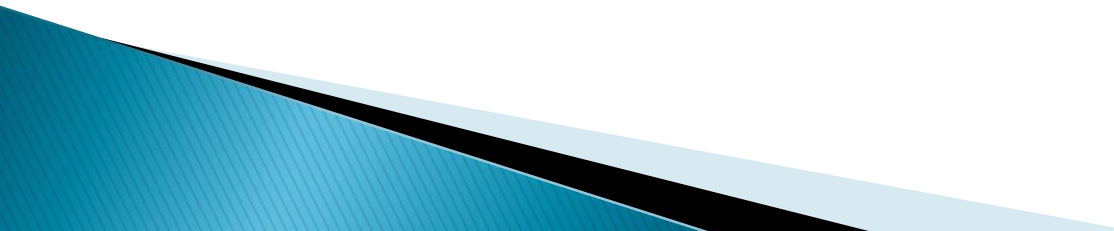


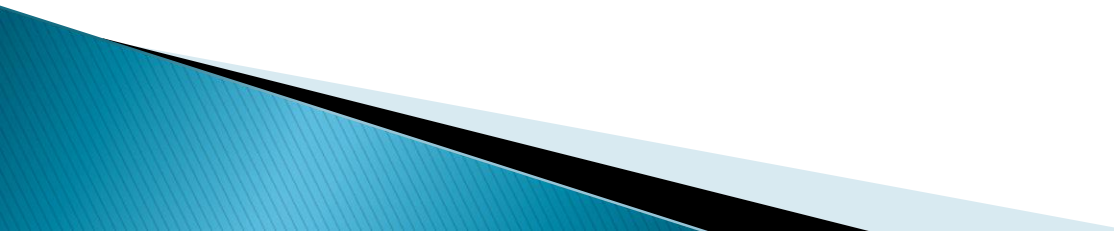
eXtreme

Programming

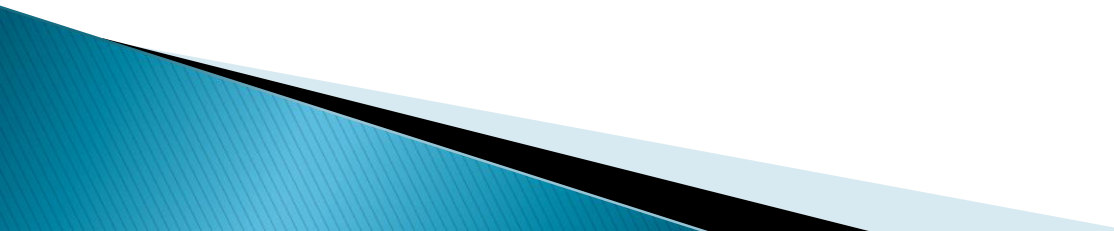


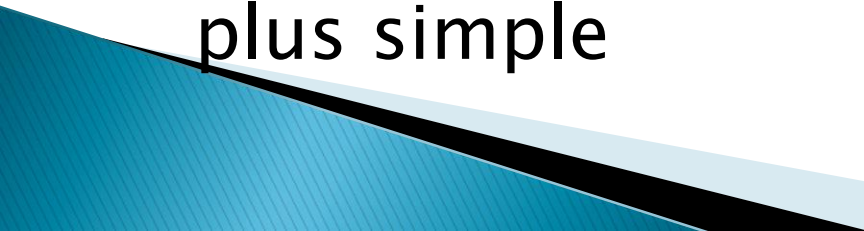
- ▶ Naissance officielle en 1999
 - ▶ XP concerne le développement logiciel
 - ▶ XP est adapté aux petites équipes
- 

XP = méthode agile

- ▶ une tentative de réconcilier l'humain avec la productivité
 - ▶ un mécanisme pour faciliter le changement social
 - ▶ une voie d'amélioration
 - ▶ un style de développement
 - ▶ une discipline de développement d'applications informatiques
- 

Rien de nouveaux :
ils existent dans l'industrie du logiciel depuis des dizaines d'années et dans les méthodes de management depuis encore plus longtemps.
L'originalité de la méthode est de les pousser à l'extrême

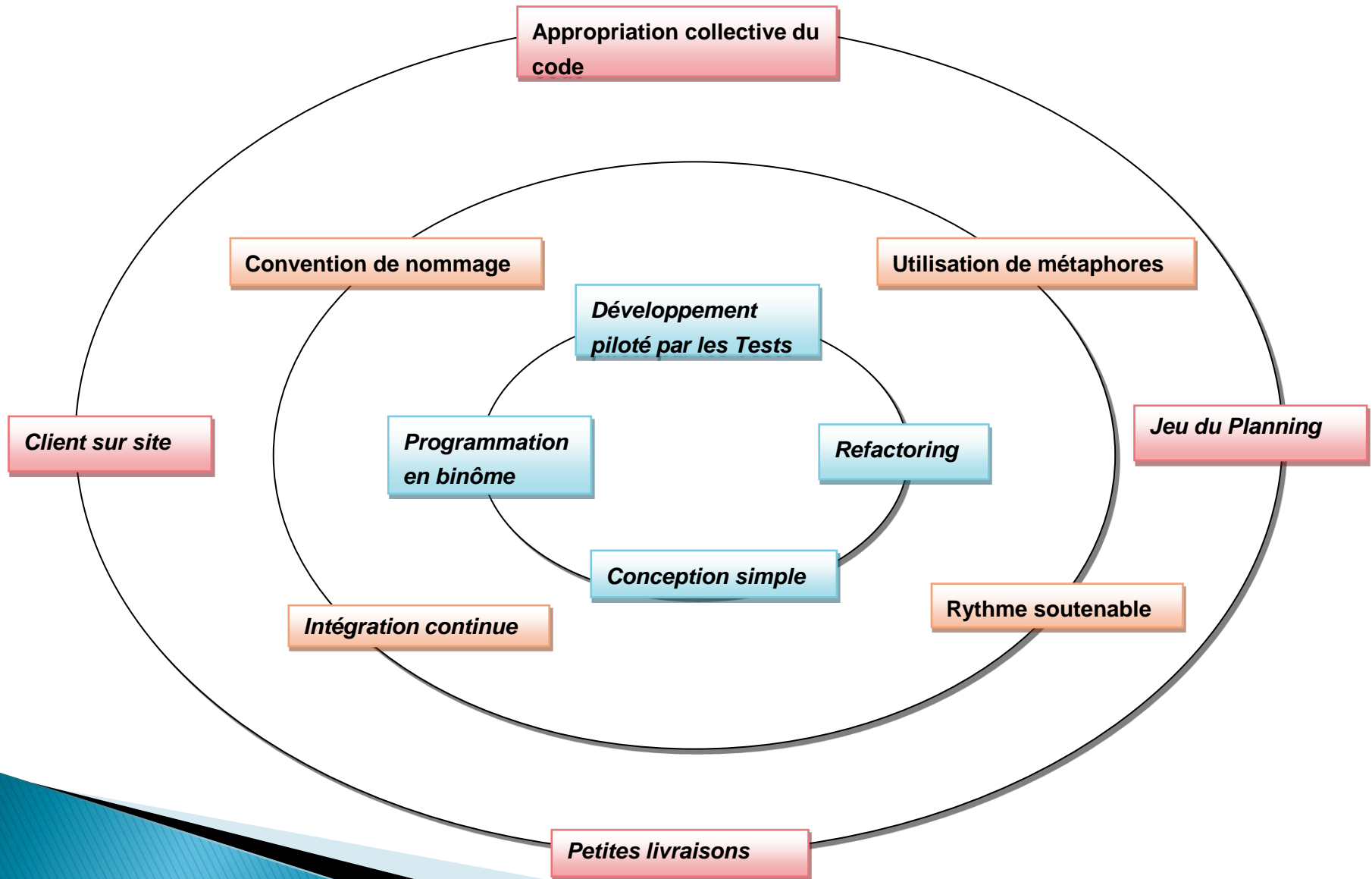


- ▶ La **revue de code** est une bonne pratique, elle sera faite en permanence (par un **binôme**)
 - ▶ les **tests** sont utiles, ils seront faits systématiquement
 - ▶ la **conception** est importante, elle sera faite tout au long du projet
 - ▶ puisque la **simplicité** permet d'avancer plus vite, nous choisirons toujours la solution la plus simple
- 

Cinq valeurs

- ▶ **La communication**
 - ▶ **La simplicité**
 - ▶ **Le *feed-back***
 - ▶ **Le courage**
 - ▶ **Le respect**
- 

Douze pratiques

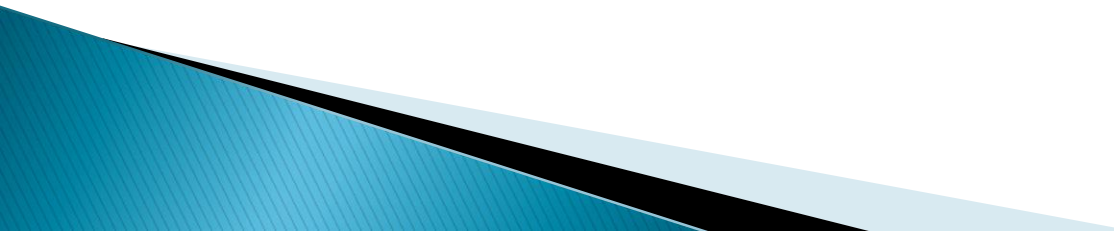


Refactoring (ou remaniement du code)

Amélioration régulière de la qualité du code sans en modifier le comportement.

On retravaille le code pour repartir sur de meilleures bases tout en gardant les mêmes fonctionnalités.

Les phases de *refactoring* n'apportent rien au client mais permettent aux développeurs d'avancer dans de meilleures conditions et donc plus vite.

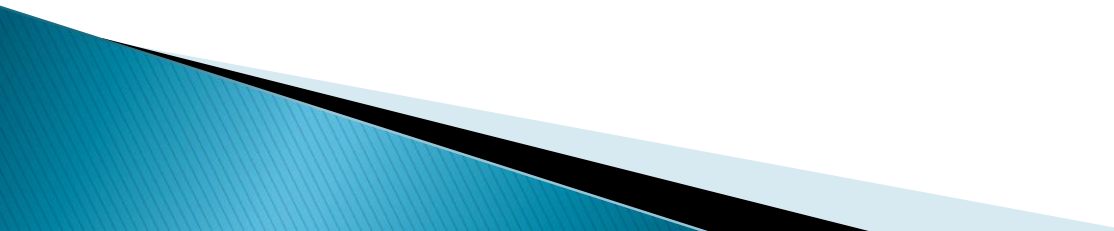


Appropriation collective du code

L'équipe est collectivement responsable de l'application.

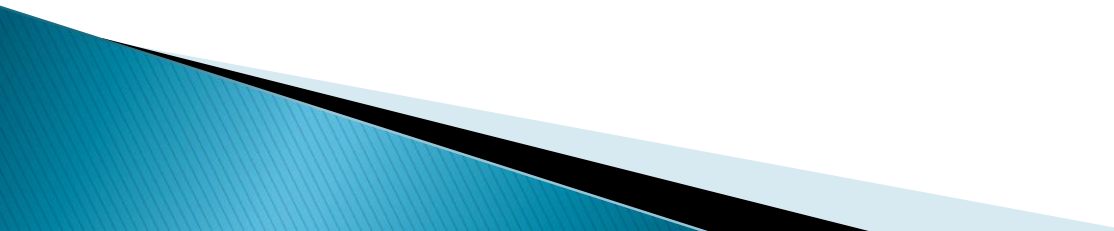
Chaque développeur peut faire des modifications dans toutes les portions du code, même celles qu'il n'a pas écrites.

Les tests diront si quelque chose ne fonctionne plus.



Convention de nommage


Puisque tous les développeurs interviennent sur tout le code, il est indispensable d'établir et de respecter des normes de nommage pour les variables, méthodes, objets, classes, fichiers, etc.



Programmation en binôme

La programmation se fait par deux. Le premier appelé *driver* (ou *pilote*) tient le clavier. C'est lui qui va travailler sur la portion de code à écrire. Le second appelé *partner* (ou *copilote*) est là pour l'aider en suggérant de nouvelles possibilités ou en décelant d'éventuels problèmes.

Les développeurs changent fréquemment de binôme ce qui permet d'améliorer la connaissance collective et d'améliorer la communication au sein de l'équipe.



Rythme soutenable

L'équipe ne fait pas d'heures supplémentaires.

Si le cas se présente, il faut revoir le planning.


Un développeur fatigué travaille mal.



Tests de recette (ou tests fonctionnels)

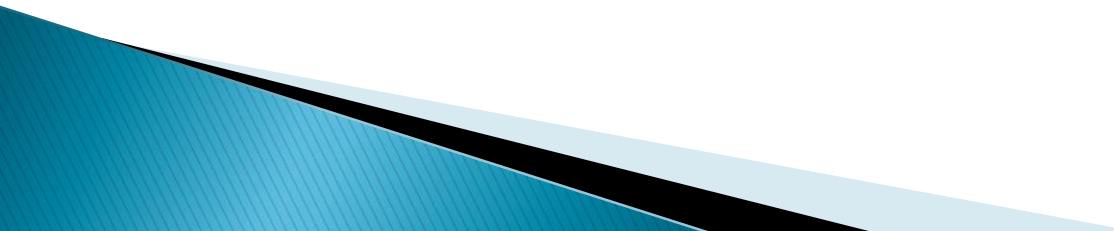
À partir des scénarios définis par le client, l'équipe crée des procédures de test qui permettent de vérifier l'avancement du développement. Lorsque tous les tests fonctionnels passent, l'itération est terminée.

La recette fonctionnelle d'une application est de plus en plus souvent confiée à des experts du test indépendants des développeurs.



Tests unitaires

Avant de mettre en œuvre une fonctionnalité, le développeur écrit un test qui vérifiera que son programme se comporte comme prévu. Ce test sera conservé jusqu'à la fin du projet, tant que la fonctionnalité est requise. À chaque modification du code, on lance tous les tests écrits par tous les développeurs, et on sait immédiatement si quelque chose ne fonctionne plus.



Conception simple

L'objectif d'une itération est de mettre en œuvre les scénarios sélectionnés par le client et uniquement cela.

Envisager les prochaines évolutions ferait perdre du temps sans avoir la garantie d'un gain ultérieur.

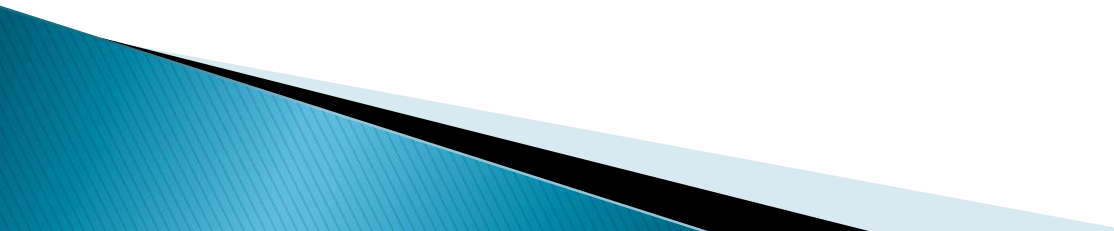
Plus l'application est simple, plus il sera facile de la faire évoluer lors des prochaines itérations.



Utilisation de métaphores

On utilise des métaphores et des analogies pour décrire le système et son fonctionnement.

Le fonctionnel et le technique se comprennent beaucoup mieux lorsqu'ils sont d'accord sur les termes qu'ils emploient.



Client sur site

Un représentant du client doit, si possible, être présent pendant toute la durée du projet.

Il doit avoir les connaissances de l'utilisateur final et avoir une vision globale du résultat à obtenir.


Il réalise son travail habituel tout en étant disponible pour répondre aux questions de l'équipe.



Jeu du Planning ou Planning poker

Le client crée des scénarios pour les fonctionnalités qu'il souhaite obtenir. L'équipe évalue le temps nécessaire pour les mettre en œuvre.

Le client sélectionne ensuite les scénarios en fonction des priorités et du temps disponible.



Intégration continue

Lorsqu'une tâche est terminée, les modifications sont immédiatement intégrées dans le produit complet.

On évite ainsi la surcharge de travail liée à l'intégration de tous les éléments avant la livraison.

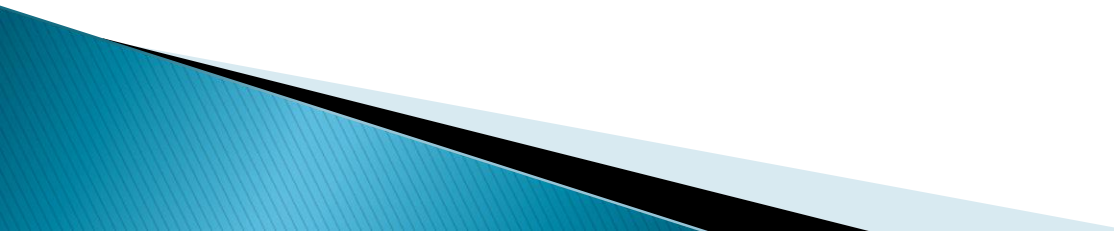
Les tests facilitent grandement cette intégration : quand tous les tests passent, l'intégration est terminée.

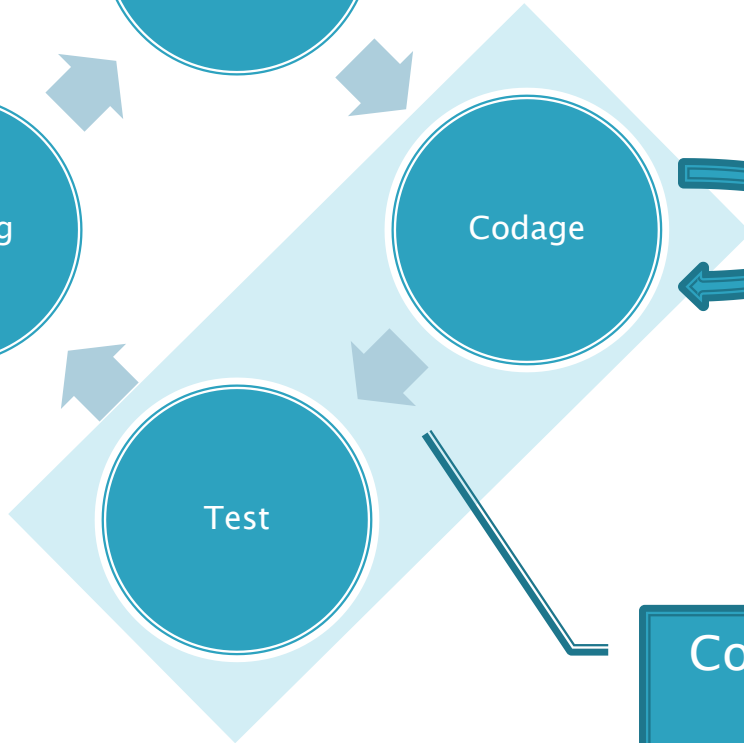
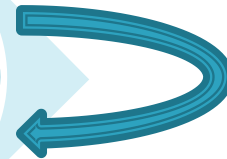
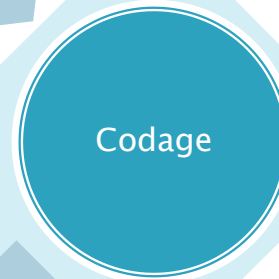
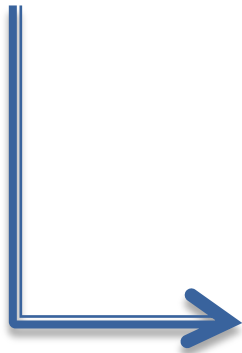


Petites livraisons

Les livraisons doivent être les plus fréquentes possible.

L'intégration continue et les tests réduisent considérablement le coût de livraison.





Codage et Test
sont
inséparables

Pour aller plus loin ...

- ▶ Méthode agile
 - ▶ Cycle de développement
 - ▶ Scrum
 - ▶ Test Driven Development
- 