XP 1999 – Kent Beck

- XP pousse à l'extrême des principes connus du management et de l'industrie du logiciel :
  - Revue du travail effectué
  - Tests du produit
  - Importance de la conception
  - Simplicité
  - Compréhension
  - Intégration
  - Evolution rapide des besoins

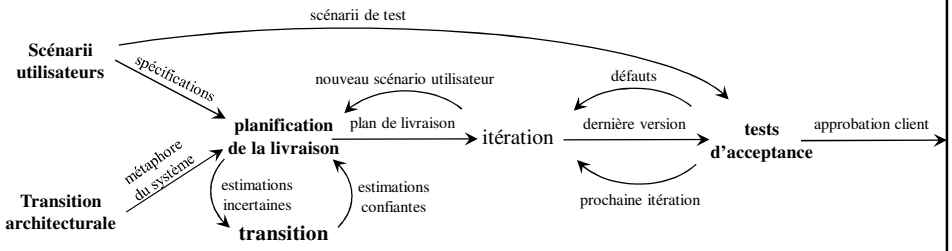
Principes de XP

- puisque la revue est une bonne pratique, elle sera faite en permanence (par un binôme) ;
- puisque les tests sont utiles, ils seront faits systématiquement avant chaque mise en œuvre ;
- puisque la conception est importante, le produit sera retravaillé tout au long du projet (refactoring) ;
- puisque la simplicité permet d'avancer plus vite, la solution la plus simple sera toujours celle qui sera retenue ;
- puisque la compréhension est importante, des métaphores seront définies et évolueront en concomitance ;
- puisque l'intégration des modifications est cruciale, celles-ci seront faites plusieurs fois par jour ;
- puisque les besoins évoluent vite, des cycles de production très rapides faciliteront l'adaptation au changement.

Cycle de XP

- XP repose sur des cycles courts (des itérations de quelques semaines, pas toutes de même durée).
- Le cycle se répète tant que le client fournit des scénarii :
  - phase d'exploration → scénarios « client » qui seront fournis pendant cette itération ;
  - l'équipe transforme les scénarios en tâches à réaliser et en tests fonctionnels ;
  - chaque équipier s'attribue des tâches et les réalise avec un binôme ;
  - lorsque le produit satisfait tous les tests fonctionnels, il est livré.

Cycle de XP



Par KrapFr based on original work of Traroth (discussion · contributions), https://commons.wikimedia.org/w/index.php?curid=29609451

5 valeurs de XP

- Communication
- Simplicité
- Retour d'information
- Courage
- Respect

Communication

- C'est le meilleur moyen d'éviter les problèmes.
- Les équipiers, les décideurs et les clients communiquent, tout le temps.
- Si un manque de communication apparaît, un coach l'identifie et remet les personnes en contact.

Simplicité

- La façon la plus simple d'arriver au résultat est la meilleure.
- Anticiper les extensions futures est une perte de temps.
- Une solution simple sera plus facile à faire évoluer.

Retour d'information

- C'est primordial pour l'équipier et le client.
- Les tests unitaires indiquent si la solution fonctionne. Les tests fonctionnels donnent l'avancement du projet.
- Les livraisons fréquentes permettent de tester les fonctionnalités rapidement.

Courage

- Certains changements demandent du courage. Il faut parfois changer l'architecture retenue, refaire certaines parties du travail ou essayer une nouvelle approche.
- Le courage permet de sortir d'une solution inadaptée. C'est difficile mais rendu accessible par les 3 premières valeurs.

Respect

- Cette valeur inclut le respect pour les autres autant que le respect de soi.
- Les équipiers ne valideront jamais des modifications qui font échouer les tests unitaires existants ou qui retardent le travail de leurs pairs.
- Les équipiers respectent leur propre travail en cherchant toujours la qualité et la meilleure conception pour la solution, et cela grâce au *refactoring*.

13 principes

- Client présent
- Poker d'estimation
- Intégration continue
- Petites livraisons
- Rythme soutenable
- Tests fonctionnels
- Tests unitaires
- Conception simple
- Utilisation de métaphores
- Remaniement
- Appropriation collective
- Travail en binôme
- Convention de nommage

Client présent

- 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.

Poker d'estimation

- 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 réussissent, 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.

Rythme soutenable

- L'équipe ne fait pas d'heures supplémentaires.
- Si le cas se présente, il faut revoir le planning.
- Un équipier fatigué travaille mal.

Tests fonctionnels

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

Tests unitaires

- Avant de satisfaire une attente, l'équipier définit un test qui vérifiera que son produit se comporte comme prévu.
- Ce test sera conservé jusqu'à la fin du projet, tant que la fonctionnalité est requise.
- À chaque modification du produit, on refait tous les tests définis par tous les équipiers, 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 la solution 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.

Remaniement

- Amélioration régulière de la qualité du produit sans en modifier le comportement, le remaniement consiste à retravailler la solution pour repartir sur de meilleures bases tout en gardant les mêmes fonctionnalités.
- Les phases de remaniement n'apportent rien au client mais permettent aux équipiers d'avancer dans de meilleures conditions et donc plus vite.

Appropriation collective

- L'équipe est collectivement responsable du produit.
- Chaque équipier peut faire des modifications dans toutes les parties de la solution, même celles qu'il n'a pas produites lui-même.
- Les tests diront si quelque chose ne fonctionne plus.

Travail en binôme

- La production se fait par deux :
  - le pilote (*driver*) réalise ;
  - le copilote (*partner*) l'aide en suggérant de nouvelles possibilités ou en décelant d'éventuels problèmes (revue permanente).
- Les équipiers changent fréquemment de partenaire et de rôle, ce qui permet d'améliorer la connaissance collective du produit et d'améliorer la communication au sein de l'équipe.

Convention de nommage

- Puisque tous les équipiers interviennent sur toutes les parties, il est indispensable d'établir et de respecter des normes de nommage pour tout (scripts, variables, méthodes, objets, classes, fichiers, etc).