Synthèse sur l’Agilité

I-

La méthode agile permet d'impliquer plus le client et offrir une meilleure

réactivité à ses demandes. L'agilité repose sur la mise en place d’une structure du cycle de développement prenant en compte la collaboration avec le client plutôt que la contractualisation des relations.

L’agilité se distingue également en favorisant le travail d'équipe, l’auto- organisation et la responsabilisation.

Les méthodes agiles décomposent les actions en petites étapes de planification. Cela permet de minimiser le risque global et de s'adapter rapidement aux changements en ayant une version définitive à chaque étape.

II -

La première des pratiques de l’eXtreme Programming, la plus fondamentale également, est le test automatisé. Seules les fonctionnalités nécessaires sont développées, et le design adopté doit être le plus simple permettant d’implémenter intégralement la fonctionnalité.

Le test automatisé n’est pas seulement une bonne pratique du développement agile : il en est l’épine dorsale (unitaires, fonctionnels, d’intégration, de performance, le plus est le mieux). Poussée à l’extrême, cette pratique a donné naissance au principe du TDD (Test Driven Development) :
- Le développeur écrit en premier lieu un test automatisé de la fonctionnalité qu’il souhaite implémenter, développe ensuite juste assez de code pour satisfaire le test et finalement remanie le code pour améliorer le design et supprimer la duplication.

XP comprend plusieurs autres principes :
- Améliorer régulièrement de la qualité du code (REFACTORING).
- L'équipe est collectivement responsable de l’application (PROPRIETE

COLLECTIVE).
- Mettre en place une convention pour nommer (NORME).
- Programmation en binôme (PAIR PROGRAMMING).
- Lorsqu'une tâche est terminée, elles est intégrées (INTEGRATION CONTINUE).

III -

Scrum peut être vu comme un framework méthodologique. Scrum définit un jeu minimal d'acteurs, de cérémonies et d'artefacts qui permettent : la planification, la gestion du temps et la gestion des obstacles.

Scrum défini 3 acteurs :

- Le Product Owner : possède l'expertise fonctionnelle et est à même de réaliser les arbitrages nécessaires à la priorisation des développements

- Scrum Master : doit optimiser la capacité de production de l'Equipe, la rendre autonome et la faire progresser.

- L'Equipe (7 à 9 personnes max) : prend en charge le développement du produit sans spécialisation des rôles. Elles est auto-organisée et dépourvue de hiérarchie.

L'unité de temps du Scrum est le Sprint. Un Sprint est une étape(itération) courte de 2 à 4 semaines.

Scrum défini également 3 artefacts : - Le Product Backlog :

-> La liste des fonctionnalités du logiciel. Les éléments du backlog sont rédigés sous forme de « User Stories ».

A minima, un élément du backlog comporte les informations suivantes : - un identifiant unique

- un nom court et descriptif

- l'estimation initiale exprimée en points de complexité (Planning Poker) - la spécification d'un test fonctionnel simple

- **des notes aussi brèves que possible.**

- **Le Product Backlog peut également contenir des éléments techniques. Il n'est pas un document figé. Au début de chaque Sprint a lieu la cérémonie qui est probablement la plus importante de Scrum : le Sprint Planning.**
  Le principal résultat de cette cérémonie est le Sprint Backlog.

- Le Sprint Backlog :

-> Résultat du Sprint Planning. Ne doit pas excéder la capacité de production de

l'Equipe.Au terme de chaque sprint, le produit partiel est livré avec le niveau de qualité attendu pour son exploitation en production.

- Le Burndown Chart :

-> La représentation graphique très simple de l'avancement du projet. Un Burndown Chart est produit pour chaque Sprint, un autre pour la version du produit.

Les principales cérémonies de Scrum sont :
- Le Sprint planning
- La mêlée quotidienne (Daily Scrum). Courte cérémonie (de l'ordre de 15 minutes) menée chaque jour.
- La revue de sprint (Sprint Review). Faire une démonstration publique du résultat du sprint.
- La rétrospective. Réunit l'équipe et le Product Owner au terme de chaque sprint afin d'identifier les erreurs commises lors du sprint précédent et de définir un plan d'actions.

IV -
Le Sprint est l'unité de temps du Scrum. Un Sprint peut être défini comme une

'étapes' permettant le développement d'une version 'finie' du projet. L'objectif étant, à la fin de chaque Sprint, de livrer une application fonctionnelle. Scrum s'appuie donc sur un développement simple de l'application et l'ajout successif de fonctionnalités. En somme, les différents Sprint permettent une intégration continue. C'est à dire, qu'ils permettent d'avoir une application rapidement fonctionnelle et fortement personnalisable par le client. Srcum, par son principe de Sprint et d'integration continue, permet de s'adapter rapidement aux demandes du clients et d'ajouter des fonctionnalités sans remettre en question l'intégrité du projet. De plus, les Sprint Planning réguliers appuie le principe d'intégration continue, tout en impliquant pleinement le client dans le projet.

La mise en place de tests permettra d'assurer l'intégrité des fonctionnalités précédemment développées.
Le Scrum permet une forme d'auto-gestion est d'implication globale de l'Equipe.

V-
Le développeur écrit en premier lieu un test automatisé de la fonctionnalité qu'il souhaite implémenter, développe ensuite juste assez de code pour satisfaire le test et finalement remanie le code pour améliorer le design et supprimer la duplication.

Il permet de maintenir le focus du développeur sur les fonctionnalités réellement utiles.

Il est définit par 3 lois :
- 1 : Vous devez écrire un test qui échoue avant de pouvoir écrire le code de production correspondant.
- 2 : Vous devez écrire une seule assertion à la fois, qui fait échouer le test ou qui échoue à la compilation.
- 3 : Vous devez écrire le minimum de code de production pour que l'assertion du test actuellement en échec soit satisfaite.

Son cycle de développement préconise 5 étapes :

1 - Ecrire un seul test qui décrit une partie du problème à résoudre .

2 - Vérifier que le test échoue, autrement dit qu'il est valide.

3 - écrire juste assez de code pour que le test réussisse.

4 - Vérifier que le test passe, ainsi que les autres tests existants.

5 - Remanier le code sans en altérer le comportement.