**S’approprier une démarche de développement agile**

**Donnez une définition de l’agilité :**

C’est une méthodologie de gestion de projets. Par exemple, on met en avant la collaboration avec le client pour améliorer la communication et ainsi répondre aux attentes plus vite et plus concrètement.

**Pourquoi l'Extreme Programming ?**

C’est une méthode de management agile permettant de faciliter le changement social. Grâce à cela, on se préoccupe davantage de l’humain qui conçoit afin de prendre en compte ses besoins et ainsi lui permettre de concevoir dans des conditions favorables, ce qui va permettre en parallèle d’avoir une finalité et un résultat plus que satisfaisant.

On cherche aussi à ce que l’équipe s’entraide en développant des méthodes agiles afin que chaque membre soit le plus performant possible sans le laisser avec des problèmes. La communication et la cohésion d’équipe font de ce programme un pilier central pour la construction du projet.

**Quels avantages y’a-t-il à utiliser SCRUM ?**

* Les projets sont trop long pour être estimés correctement.
* Scrum permet d’être transparent envers les métiers.
* La flexibilité des développements est permise grâce aux itérations et aux changements de priorités du product backlog.
* L’agilité favorise la motivation et l’auto-organisation des équipes de développement.
* Les utilisateurs communiquent directement avec l’équipe et réduisent les risques d’incompréhensions.
* Un sentiment de reconnaissance des équipes se fait immédiatement sentir.

**Sprint et intégration continue, expliquez le fonctionnement de SCRUM**

C’est une méthode de conduite de projets basée sur des valeurs et principes agiles. Scrum définit un jeu d'activités qui permettent aux clients d'étudier, de guider et d'influencer le travail de l’équipe de développement au cours de sa progression. L’équipe travaille par courtes itérations (appelées aussi sprints) et affine son plan à mesure qu'elle avance dans son travail. Ces différents sprints ont une durée fixe, toujours identique, et fixée à l’origine du projet. Cela permet de connaître la vélocité de l’équipe et de pouvoir être toujours plus précis dans l’estimation du temps de réalisation d’une partie de l’application.

**TDD :(Test Driven Development), recherchez des informations pour expliquer le principe.**

C’est une méthode de [développement de logiciel](https://fr.wikipedia.org/wiki/D%C3%A9veloppement_de_logiciel), qui consiste à concevoir un [logiciel](https://fr.wikipedia.org/wiki/Logiciel) par petits pas, de façon itérative et incrémentale, en écrivant chaque test avant d'écrire le [code source](https://fr.wikipedia.org/wiki/Code_source) et en remaniant le code continuellement.

Trois lois se dégagent de cette méthode

1. Loi no 1 : "Vous ne pouvez pas écrire de code de production tant que vous n'avez pas écrit un test unitaire qui échoue." ("You may not write production code until you have written a failing unit test.").

C'est-à-dire **qu'il n'est permis d'écrire du code de production que si un test unitaire est en échec**.

1. Loi no 2 : "Vous ne pouvez pas écrire plus d'un test unitaire que nécessaire pour échouer, et ne pas compiler revient à échouer." ("You may not write more of a unit test than is sufficient to fail, and not compiling is failing.").

C'est-à-dire **qu'il n'est permis d'écrire qu'un nouveau test unitaire en échec à la fois, et un test unitaire qui ne compile pas est déjà un test en échec**.

1. Loi no 3 : "Vous ne pouvez pas écrire plus de code de production que nécessaire pour que le test unitaire actuellement en échec réussisse." ("You may not write more production code than is sufficient to pass the currently failing test.")

C'est-à-dire **qu'il n'est permis d'écrire que du code de production permettant directement de faire passer le test unitaire précédent, ni plus ni moins**.

Le TDD permet d'explorer et de préciser le besoin, puis de spécifier le comportement souhaité du logiciel en fonction de son utilisation, avant chaque étape de codage. Le logiciel ainsi produit est tout à la fois pensé pour répondre avec justesse au besoin et conçu pour le faire avec une [complexité minimale](https://fr.wikipedia.org/wiki/Complexit%C3%A9_essentielle). On obtient donc un logiciel mieux conçu, mieux testé et plus fiable, autrement dit de meilleure qualité.