

## Programmation Agile : Principes et techniques du développement réussi de logiciels - 3 jours formation 931

- Vous apprendrez à**
- Développer des itérations plus rapidement en utilisant des méthodologies agiles telles que XP et Scrum
  - Collecter des scénarios, estimer la charge de travail et planifier les itérations, les sprints et les releases (versions)
  - Réduire les bogues et augmenter la productivité grâce au développement piloté par les tests et grâce aux tests unitaires
  - Refactoriser le code existant pour faciliter la maintenance et améliorer la conception
  - Élaborer des conceptions de qualité en adoptant des principes de codage établis et des patterns
  - Faciliter la communication au sein de l'équipe avec des réunions quotidiennes, des revues d'itération et des rétrospectives

**Objectif** Économiser du temps et de l'argent sur le développement de logiciels est essentiel pour rester compétitif. Les méthodologies agiles, notamment Scrum et XP, réduisent les coûts, augmentent la qualité et accélèrent le temps de commercialisation. Cette formation vous permet de maîtriser les bases des principes de la programmation agile. Grâce à une étude de cas complète, vous acquerez les connaissances et les compétences pratiques nécessaires pour planifier, coder et mettre en œuvre un projet de développement logiciel agile.

**À qui s'adresse cette formation** À toute personne travaillant au sein d'une équipe de développement de logiciel, y compris les analystes, les concepteurs, les programmeurs, les testeurs et les cadres techniques. Une compréhension des pratiques élémentaires en ingénierie logicielle et une expérience dans le développement sont supposées acquises.

**RealityPlus** Tout au long de la formation, des activités de mise en situation et sur PC vous immergeront dans la simulation d'un authentique projet de programmation agile. Vous effectuerez des tâches critiques, notamment :

- Interviewer des clients pour générer des scénarios utilisateur
- Estimer les scénarios utilisateur et leur attribuer des priorités
- Participer à une "spike session" pour découvrir une nouvelle technologie
- Écrire des tests et du code pour concrétiser les scénarios utilisateur
- Refactoriser le code pour éliminer le code douteux afin d'obtenir une conception élégante
- Mettre en œuvre une architecture adaptable en employant des design patterns
- Livrer un logiciel par itérations fréquentes
- Mener des revues et des rétrospectives de sprint pour faciliter l'apprentissage

## Programmation Agile : Principes et techniques du développement réussi de logiciels - 3 jours formation 931

### Introduction et vue d'ensemble

- Adopter les bonnes pratiques du Manifeste Agile
- Comparer les méthodologies de développement logiciel pilotées par la planification et agiles
- Identifier les bonnes pratiques agiles et les étapes d'un processus agile

### Planification d'une release agile

#### Établir le projet

- Reconnaître la structure d'une équipe agile
- Programmeurs
- Managers
- Clients
- Créer des équipes transversales

#### Développer une base avec des scénarios utilisateur

- Expliciter les exigences de l'application
- Recueillir les scénarios utilisateur
- Reconnaître les bons scénarios utilisateur

#### Estimer

- Définir une unité d'estimation
- Estimer la charge de travail relative
- Calculer la vélocité de l'équipe
- Redéfinir le budget en fonction de la vélocité

#### Planifier des releases, itérations et sprints

- Le "planning game"
- Établir un budget d'itération
- Donner la priorité aux scénarios
- Créer des backlogs de produit et de sprint
- Gérer les dérapages par rapport aux objectifs

#### Comparer Scrum et XP

- Scrum Master
- Directeur de produit
- Équipe Scrum

#### Produire des logiciels adaptatifs grâce au développement piloté par les tests

#### Piloter le processus de conception avec des tests automatisés

- Programmer des tests automatisés
- Valider les scénarios et les tests d'acceptation
- Écrire le code en fonction du résultat des tests

### Intégrer les tests unitaires

- Développer des suites de tests efficaces
- Concevoir un test unitaire simple, isolé et rapide
- Obtenir des "feux verts" grâce aux tests continus
- Isoler des classes pour tester efficacement
- Créer des objets factices pour tester

### Refactoriser pour une conception élégante

#### Flairer les symptômes de dysfonctionnement du code

- Logique conditionnelle
- Duplication de code
- Code nécessitant des commentaires

#### Améliorer le code par refactorisation

- Renommage des champs et des méthodes
- Extraction des méthodes et des classes de base
- Concevoir un code simple et réutilisable

### Intégrer les principes de la programmation orientée objet

#### Adopter les principes des bonnes pratiques

- Le principe de responsabilité unique (SRP, Single Responsibility Principle)
- Le principe d'ouverture-fermeture (OCP, Open/Closed Principle)
- Le principe d'inversion des dépendances (DIP, Dependency Inversion Principle)
- Le principe de substitution de Liskov (LSP, Liskov Substitution Principle)
- Principe de ségrégation des interfaces (ISP, Interface Segregation Principle)

#### Faire évoluer la conception grâce à la généralisation

- Déléguer les responsabilités des classes
- Mettre en œuvre des comportements polymorphiques
- Préférer la composition à l'héritage

### Simplifier des problèmes d'architecture complexes avec des design patterns

#### Présentation des design patterns

- Patterns créateurs, structuraux et comportementaux
- Garantir la correction d'une conception adaptable

### Intégrer des design patterns aux processus agiles

- Patron de méthode
- Adaptateur
- Stratégie
- Singleton
- Fabrication

### Finir une itération

#### Revoir une itération ou un sprint

- Livrer le logiciel
- Estimer les itérations ultérieures
- Ajuster le budget selon la vélocité actuelle

#### Mener une rétrospective de sprint

- Brainstorming pour récapituler ce qui a été appris
- Créer un plan d'action pour maximiser la productivité
- Incorporer les bonnes pratiques et les outils agiles