

Conception de logiciels orientés objet - 4 jours

formation 1801

- Vous apprendrez à**
- Livrer des logiciels respectant le planning et le budget, en utilisant des démarches itératives et agiles
 - Recueillir efficacement les besoins en utilisant des récits utilisateurs développés sous forme de cas d'utilisation
 - Modéliser intelligemment avec UML pour enrichir le processus de conception
 - Concevoir des architectures orientées objet hautement réutilisables à base de composants
 - Produire des systèmes logiciels souples et adaptables en utilisant une conception itérative et incrémentale
 - Garantir des implémentations robustes grâce au développement piloté par les tests, aux patrons de conception (design patterns) et à la restructuration du code

Objectif Dans un environnement professionnel en constante évolution, produire rapidement un logiciel ouvert, capable de s'adapter aux évolutions technologiques et aux changements des besoins utilisateur constitue un réel avantage compétitif. Associer la modélisation UML aux démarches agiles est une approche qui a fait ses preuves pour développer de tels logiciels. Au cours de cette formation, vous apprendrez à analyser, concevoir et réaliser des logiciels en utilisant des méthodes itératives et incrémentales particulièrement efficaces.

À qui s'adresse cette formation Aux développeurs et aux concepteurs de logiciels, responsables d'équipes, chefs de projets et aux analystes en recueil des besoins. Des connaissances des concepts orientés objet sont supposées acquises.

Travaux pratiques Les travaux pratiques vous permettent de vous familiariser avec le langage UML et les approches agiles, itératives et incrémentales. Les exercices et les démonstrations comprennent :

- Développer les récits d'utilisateurs en cas d'utilisation
- Décrire le comportement des cas d'utilisation en utilisant les diagrammes de séquence et d'activité UML
- Modéliser un comportement complexe avec des diagrammes états-transitions
- Construire une architecture statique en utilisant des diagrammes de classe et de composants
- Produire et améliorer du code en utilisant la technique des développements pilotés par les tests (Test Driven Development ou TDD)
- Extraire et identifier dans le code des patrons de conception (design patterns)

Conception de logiciels orientés objet - 4 jours

formation 1801

Introduction

- Adapter la méthode à l'échelle du projet
- Démarche agile dans le cadre d'un développement itératif
- Modéliser efficacement la conception avec UML
- Implémenter la conception en utilisant les techniques de développement piloté par les tests

Adapter la méthode au projet

Évaluer les approches traditionnelles

- Faire la critique des cycles de vie en cascade et en V
- Répondre au changement de façon itérative et incrémentale

Explorer les solutions itératives et agiles

- Identifier les risques des démarches purement agiles
- Réduire le risque avec une conception basée sur UML

Recueillir fidèlement les besoins

Se préparer au développement itératif et incrémental

- Identifier et catégoriser les parties prenantes
- Recueillir les récits d'utilisateurs et remplir le "backlog"
- Affiner les besoins en développant les récits en cas d'utilisation

Planifier un cycle itératif

- Estimer le travail de conception et de développement à partir des récits d'utilisateurs
- Intégrer les priorités des parties prenantes
- Traiter les récits utilisateurs incomplets ou dépendants

Concevoir efficacement les récits utilisateurs grâce à UML

Modéliser juste ce qu'il faut

- Modéliser la structure statique : diagrammes de classes et de composants
- Éviter de sur- ou sous-modéliser
- Représenter le comportement des cas d'utilisation avec des diagrammes d'activité

Concevoir l'architecture dynamique

- Modéliser les cas d'utilisation en trois tiers
- Diagrammes de séquence illustrant le comportement des cas
- Contrôler les scénarios alternatifs avec les diagrammes états-transitions UML

- Implémenter le comportement des cas d'utilisation sous forme d'architecture Modèle-Vue-Contrôleur (MVC)

Représenter l'architecture statique

- Préparer un modèle d'entité en utilisant classes et associations
- Valider la structure des données en la confrontant au modèle dynamique

Ingénierie du logiciel

Documenter la conception détaillée avec UML

- Construire les diagrammes de classes de réalisation
- Décrire le comportement du code avec des diagrammes de séquence et d'états-transitions
- Associer les modèles de l'outil de génie logiciel aux livrables de l'itération

Appliquer les bonnes pratiques de pilotage par les tests

- Rédiger des tests de récits d'utilisateurs et de cas d'utilisation démontrables
- Choisir les bons tests unitaires : tableaux d'équivalence et valeurs limites
- Automatiser les processus des tests avec des tests unitaires et des structures simulées

Restructurer pour optimiser le logiciel

- Améliorer la réutilisation en appliquant le principe ouvert-fermé
- Réduire le couplage et augmenter la cohérence grâce à la responsabilité unique
- Extraire les interfaces et gérer l'injection de dépendance

Augmenter la modularité grâce aux patrons de conception

- Découpler le comportement grâce aux patrons Stratégie et Décorateur
- Isoler les couches d'un modèle 3-tiers grâce aux patrons MVC et Observateur
- Centraliser la création d'objets avec des fabriques de classes

Intégrer des sous-systèmes pour créer un système fonctionnel

- Partitionner les tests en simulant les interfaces en aval
- Construire et exécuter les tests d'intégration
- Minimiser le couplage avec les patrons Façade et Proxy

Stabiliser le processus itératif

Mener une itération à son terme

- Valider des récits d'utilisateurs et des cas d'utilisation complets
- Valoriser l'itération : comparer les budgets réels et estimés
- Ajuster le processus pour les prochaines itérations

Acquérir les bons outils

- Comparer les outils de tests automatisés
- Suivre le planning et l'organisation avec des outils de gestion de projet
- Reproduire le contrôle des versions pour les besoins, la modélisation et le code