3IF2225 - Génie Logiciel

Responsables : **Paolo BALLARINI**

Langues d'enseignement : **FRANCAIS**

Campus où le cours est proposé : **CAMPUS DE PARIS - SACLAY**

Nombre d'heures d'études élèves (HEE) : **30**

Nombre d'heures présentielles d'enseignement (HPE) : **15**

Année académique : **2024-2025**

Niveau avancé : **non**

Présentation, objectifs généraux du cours :

Ce cours a pour principal objectif de fournir aux étudiants en ingénierie une vue d'ensemble sur les différentes approches, techniques et méthodes utilisées dans la réalisation des logiciels critiques (à fort impact sur l'économie, la vie humaine, ...) et de grande taille (mobilisant des ressources considérables). Il doit permettre à l'apprenant de :

- Comprendre ce que c'est le Génie Logiciel ainsi que ses objectifs,
- Connaître les différentes approches de développement logiciel,
- Percevoir de façon générique le processus unifié et ses caractéristiques,
- Maîtriser les différents enchaînements d'activités du processus unifié,
- Mettre en œuvre le processus unifié dans le cadre d'un projet.

Le cours va se concentrer sur le paradigme de la programmation orientée objet (POO) ainsi que sur l'application des formalismes de modélisation comme les diagrammes de classes et des séquences UML. Un accent particulier est mis sur des aspects tels que la réutilisabilité et l'adaptibilité du code à travers l'application des patrons de conception.

Période(s) du cours (n° de séquence ou hors séquence) :

SM11

Prérequis :

aucun

Plan détaillé du cours (contenu) :

Notions de base du génie logiciel et de la qualité des logiciels.

- modèles de développement logiciel,
- facteurs de qualité (externe vs interne),
- principes et critères de modularité,

- Cycles de conception
  - cycle en V,
  - expression des besoins,
  - spécification, implémentation, vérification, validation
- Paradigme de la programmation orientée objet
  - Les langages de programmation en tant qu'abstractions du code machine.
  - Évolution de l’abstraction. Notion d’objet (état et comportement) et type d’objet.
  - Le calcul comme échange de messages entre objets.
  - Expression des problèmes en termes d’objets.
  - Structure des programmes de POO.
  - La POO comme paradigme naturel pour atteindre la qualité du logiciel.
  - POO en JAVA: classes et éléments de base du language JAVA.
- Pratiques pour la qualité des logiciels en POO.
  - Réutilisation du code.
  - Masquage de l’information. Encapsulation.
  - Composition de classes vs. héritage.
  - Notions liées à l’héritage : shadowing, overriding/hiding, polymorphisme. Classes abstraites. Interfaces.
- Tests de code et développement piloté par les tests (TDD).
  - Introduction et motivation : en quoi consistent les tests de code et pourquoi ils sont utiles.
  - Écrire des tests unitaires avec Junit.
  - Introduction à la méthodologie de développement piloté par les tests (TDD) : des unités de test qui échouent aux unités de test qui réussissent.
- Diagrammes de classe et diagrammes de séquence UML.
  - Rôle de la modélisation dans le développement de la POO.
  - Diagrammes de classes. Relations. Annotations de relations.
  - Association générique, agrégation, composition.
  - Relations entre classes : généralisation, implémentation.
  - Diagrammes de séquences: modélisation des interactions des processus disposés dans l’ordre chronologique.
- Patrons de conception.
  - Introduction et motivation.
  - Le principe d’ouverture et de fermeture.
  - Familles de patrons : patrons de conception créatifs, comportementaux et structurels.
  - Étude et application de quelques Patron de conception.

Déroulement, organisation du cours :

Cours magistraux pour présenter les concepts

- Séances pratiques avec réalisations concrètes pour mettre en œuvre les concepts et se les approprier

Organisation de l'évaluation :

L'évaluation se fera en contrôle continu sur la qualité du travail fourni et des rendus sur un mini-projet avec patrons.

Moyens :

Les moyens mis en œuvre pour ce cours combinent cours magistraux et séances pratiques favorisant l'assimilation des notions présentées.

Acquis d'apprentissage visés dans le cours :

À l'issue de ce cours, les élèves seront capables :

- de développer des solutions standard de POO à partir d’un ensemble de spécifications/exigences,
- de prendre en compte les aspects de qualité du logiciel lors de la conception d’une solution logicielle,
- de distinguer les choix de conception qui aboutissent à des solutions plus flexibles de ceux qui aboutissent à un code dont l’extension est très coûteuse.

Description des compétences acquises à l'issue du cours :

C1.4 Specify, design, implement and validate all or part of a complex system

Define the structure of a software system
Design software taking into account its life cycle
C2.1 Have a deep understanding of a basic science or engineering field or discipline

Knowledge of standard software system modelling practices.
C6.4 Solve problems using computational thinking

Be able to model a software solution.

Bibliographie :

- "Design Patterns: Elements of Reusable Object-Oriented Software". Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides.
- "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development". Craig Larman