Processus de développement en ingénierie logicielle

Domaine Ingénierie et Architecture
Filière Informatique et systèmes de communication
Orientation Informatique logicielle (ISCL)
Mode de formation Plein temps

Informations générales

Nom : Processus de développement en ingénierie logicielle
Identifiant : DIL
Années académiques : 2021-2022, 2022-2023
Responsable : Bertil Chapuis
Charge de travail : 90 heures d'études
Périodes encadrées : 64 (= 48 heures)

Semestre	E1	S1	S2	E2	S3	S4	E3	S5	S6
Cours						32			
Laboratoire						32			

Connaissances préalables recommandées

L'étudiant-e doit connaître et savoir utiliser les notions suivantes :

- connaissances générales d'un langage de programmation (C, C++, Java) ;
- connaissances en programmation orientée objet (classe, classe abstraite, interface, héritage...) ;
- capacité à modéliser un problème.

Les unités d'enseignement PRG1, PRG2 (informatique) et POO (programmation orientée objet) permettent d'acquérir ces connaissances.

Objectifs

A l'issue de cette unité d'enseignement, l'étudiant-e sera capable de :

- expliquer les enjeux du processus de développement en ingénierie logicielle ;
- décrire les principaux modèles de cycle de vie du logiciel (en cascade, agile, etc.) et estimer leurs avantages et inconvénients respectifs ;
- expliquer les concepts-clés de la programmation extrême (XP) et du processus unifié (UP) ;
- spécifier les besoins utilisateur (les fonctionnalités attendues du système à développer) en termes de scénarios et de diagrammes de cas d'utilisation ;
- décrire les différents diagrammes (statiques et dynamiques) UML et appliquer judicieusement ces derniers dans les diverses phases du développement d'un projet (notamment la spécification, l'analyse et conception) ;
- estimer le temps de manière plus précise à l'aide d'outils de planification tels que CPM (critical path method) et PERT (program evaluation and review technic) ;
- appréhender la problématique des tests en général et des tests unitaires en particulier ;
- décrire les principales mesures de qualité propres à l'ingénierie logicielle (analyse statique du code, analyse des dépendances, couverture du code par les tests, etc.) et appliquer ces mesures à l'analyse concrète d'un code ;
- appréhender la problématique de la mesure de performance avec le profiling et le benchmarking.

A l'issue des travaux pratiques en laboratoire, l'étudiant-e sera capable de :

- procéder à la spécification, l'analyse et la conception orientée objet d'un logiciel ;
- comprendre et mettre en œuvre les différents diagrammes UML ;
- planifier un développement de projet en s'appuyant sur une méthode agile ;
- concevoir des tests unitaires et d'intégration pertinents ;
- mesurer la qualité d'un logiciel et ses performances.

Contenu et formes d'enseignement

Répartition des périodes indiquée à titre informatif.

Cours: 32 périodes

- Introduction à l'ingénierie logicielle : définition, crise du logiciel, contraintes, complexité	2
- Processus de développement : définition, activités, modèle en cascade, modèle incrémental	2
- Processus agiles : Présentation des idées-clés de la programmation extrême (XP) et du processus de développement unifié (UP)	4
- Spécification: Ecriture de scénarios et diagrammes de cas d'utilisation	6
- Conception: Diagrammes de classes, d'objets et de composants	2
- Conception: Diagrammes de séquence	2
- Conception: Diagrammes d'activité	2
- Conception: Diagrammes de machine à états	2
- Estimation: CPM (critical path method) et PERT (program evaluation and review technic)	2
- Tests: Types de tests (unitaires, intégration, système) et exemples	4
- Qualité: Analyse statique du code, des dépendances, de la couverture des tests	2
- Performance: Profiling et benchmarking	2

Laboratoire: 32 périodes

- Ecriture de scénarios	4
- Diagrammes de cas d'utilisation	2
- Diagrammes de classes, d'objets et de composants	2
- Diagrammes de séquence	2
- Diagrammes d'activité	2
- Diagrammes de machine à états	2
- Gestion de la configuration et des dépendances avec maven	2
- Intégration continue et automatisation	4
- Tests unitaires et d'intégration avec JUnit 5	4
- Analyse des dépendances, analyse de la couverture des tests, analyse statique du code	4
- Mesure de la performance avec Flight Recorder (profiling) et JMH (benchmarking)	4

Bibliographie

- A Philosophy of Software Design, John K. Ousterhout, Yaknyam Press, 2018.
- Software Engineering, Ian Sommerville, Pearson, 2016.
- Learning Agile: Understanding Scrum, XP, Lean, and Kanban, par Andrew Stellman et Jennifer Greene, O'Reilly, 2014.
- UML distilled, par Martin Fowler, Addison Wesley, 2013.
- Applying UML and Design patterns, par Craig Larman, Prentice Hall, 2004.
- Object Oriented Modeling & Design with UML, par R. Blaha & James Rumbaugh, Pearson, 2004.
- RUP, XP, architectures et outils : Industrialiser le processus de développement, par Pierre-Yves Cloux, 2003 Dunod.

Contrôle de connaissances

Cours : l'acquisition des matières de cet enseignement sera contrôlée au fur et à mesure par des tests et des travaux personnels tout au long de son déroulement. Il y aura au moins 3 tests d'une durée totale d'au moins 2 périodes.

Laboratoire : ils seront évalués sur la base des rapports de manipulation, à 3 reprises au minimum.

Calcul de la note finale

Note finale = moyenne cours x 0.67 + moyenne laboratoire x 0.33