Génie logiciel

Domaine	Ingénierie et Architecture
Filière	Informatique
Orientation	Informatique embarquée (IE)
Mode de formation	Temps partiel/En emploi

Informations générales

Nom:	:	Génie logiciel
Identifiant:	:	GEN
Années académiques	:	2016-2017, 2017-2018, 2018-2019, 2019-2020
Responsable:	:	Olivier Cuisenaire
Charge de travail:	:	150 heures d'études
Périodes encadrées:	:	96 (= 72 heures)

Semestre	E1	S1 S2	E2	S3 S4	E3	S5 S6	E4	S7 S8
Cours				48				
Laboratoire				48				

Connaissances préalables recommandées

L'étudiant doit maîtriser les concepts étudiés dans les unités d'enseignement INF1 et INF2.

Il doit également connaître les principaux concepts de la programmation orientée objet (classe, classe abstraite, interface, héritage…) et avoir de bonnes connaissances du langage Java (l'unité d'enseignement POO permet d'acquérir ces connaissances).

Objectifs

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

- Expliquer les enjeux de la démarche "génie logiciel"
- Décrire les principaux modèles de cycle de vie du logiciel, estimer leurs avantages et inconvénients respectifs
- Spécifier les besoins utilisateur (les fonctionnalités attendues du système à développer) en termes de diagrammes de cas d'utilisation
- Expliquer à quoi sert un outil de configuration et quels en sont les principes de fonctionnement
- Appréhender la problématique des tests en général et des tests unitaires en particulier
- 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).
- Expliquer les possibilités offertes par les quelques design patterns étudiés au cours
- Expliquer les concepts-clés de la programmation extrême et du processus unifié
- Décrire les principes des métriques standards étudiées, appliquer ces métriques à l'analyse concrète d'un code

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

- Utiliser un outil UML
- Comprendre et mettre en œuvre les différents diagrammes UML
- Procéder à la spécification, l'analyse et la conception orientée objet d'un logiciel au moyen d'UML
- Planifier un développement de projet en s'appuyant sur un cycle de vie itératif incrémental de type UP
- Utiliser un outil comme GIT pour le travail collaboratif et la gestion de version
- Concevoir des tests unitaires pertinents

Contenu et formes d'enseignement

Répartition des périodes indiquée à titre informatif.

Cours: 48 périodes

- Introduction au génie logiciel: sensibilisation, crise du logiciel, définition, objectifs, exemples	4
- Processus de développement : - Introduction : définitions, gestion du risque, documentation - Cycle de vie du logiciel : modèle en cascade, modèle en spirale, modèle itératif incrémental	8
- Processus agiles : - Présentation des idées-clés de la programmation extrême et du processus de développement dit "Processus unifié (UP)" - Discussion des différences entre les deux approches	8
- Spécification : - Introduction : objectif, spécification formelle et semi-formelle - Cas d'utilisation d'UML : exigence, rôle des cas d'utilisation par rapport au processus de développement, acteurs et niveaux d'objectifs, scénario nominal et secondaire, portée, conditions et garanties de succès, sous-cas, extension de cas, rôle du cas d'utilisation par rapport à la conception	8
- Conception : - Statique : diagrammes (UML) de classes, d'objets, de composants, de déploiement - Dynamique : diagrammes de séquence, de collaboration, d'états, d'activités - Design patterns : enjeu, présentation des patterns classiques MVC & Modèle en couche	12
- Gestion des configurations : - Introduction : gestion du changement, travail en équipe, processus, outils - Configuration : identification des configurations, notion de référentiel et d'espace de travail, gestion de versions et de branches - Outils: présentation d'un outil-type du marché	4
- Tests : - Introduction : exemples d'erreurs et de tests typiques, types de tests, étapes du test - Activités de test : planification, conception, procédure de test, évaluation des résultats - Test unitaire : JUnit	4

Laboratoire: 48 périodes

- Spécification avec modélisation UML (cas d'utilisation & diagramme de séquence)	4
- Conception avec modélisation UML (diagrammes statiques & dynamiques)	6
- Structuration MVC & Modèle Observable-Observé	6
- Gestion de code avec Ant	2
- Gestion de version et travail collaboratif avec GIT	4
- Tests unitaires avec Junit	4
- Gestion d'un développement logiciel selon UP (phases d'élaboration et de construction, application à un mini-projet de groupe)	22

Bibliographie

Software Engineering, par Ian Sommerville, Addison Wesley, 2010

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

Learning Agile: Understanding Scrum, XP, Lean, and Kanban, par Andrew Stellman et Jennifer Greene, O'Reilly, 2014

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 2 tests d'une durée totale d'au moins 3 périodes.

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

Examen:
L'atteinte de l'ensemble des objectifs de formation sera vérifiée lors d'un contrôle final commun écrit d'une durée de 90 minutes.

Matériel autorisé:

- Information communiquée directement par l'enseignant.

Calcul de la note finale

Note finale = moyenne cours x 0.3 + moyenne laboratoire x 0.2 + moyenne examen x 0.5

Fiche validée le 16.08.2016 par Donini Pier