NFP209 - Construction rigoureuse des logiciels(1)

Valide le 22-03-2019

Code : NFP209

6 crédits

Responsabilité nationale :
EPN05 - Informatique / Nicole LEVY

Contact national :
EPN05 - Informatique
2 rue Conté
33.1.13A
75003 Paris
01 40 27 26 81
Safia Sider
safia.sider@lecnam.net

Présentation

Prérequis

Voir les UE GLG 206 et GLG 207
Public :
Informaticiens désireux d'acquérir une formation dans le domaine des Logiciels Sûrs. Étudiants visant le diplôme d'ingénieur Cnam spécialité Informatique pour les parcours IRSM et AISL.
Prérequis :
Bonne connaissance d'un langage de programmation. Il est conseillé d'avoir suivi ou de suivre en parallèle la valeur : Spécification et Modélisation Informatiques (SMI) (code NFP 108).

Objectifs pédagogiques

Dans de nombreuses applications comme les télécommunications, les transports terrestre et aérien, le nucléaire, les banques, les soins médicaux etc, un logiciel défectueux peut conduire à des défaillances aux conséquences irréversibles, voire dramatiques.
En dehors de ces domaines critiques, l'essor d'internet, la généralisation de l'utilisation de langages comme JAVA permettant la mobilité du code, accroît et généralise le besoin de sécurité et par là même la nécessité de la vérification. D'où la profusion actuelle de projets et d'outils tant industriels qu'universitaires autour de la validation et de la vérification de code.
De nombreuses entreprises travaillant sur ces sujets nous ont fait part de leurs difficultés à recruter des ingénieurs ayant une formation adaptée. C'est pour répondre aux besoins exprimés par ces entreprises, que nous avons mis en place, en collaboration avec elles, un enseignement dans ce domaine.
L'UE NFP209 fournit les techniques de description et d'analyse utilisées dans les méthodes visant à accroître la robustesse, la sûreté et la performance des langages et des logiciels. Cette UE consiste en un panorama de techniques et d'outils de validation de logiciels.
NFP209 fait aussi partie intégrante diplôme d'ingénieur du CNAM, spécialité Informatique pour les parcours IRSM et AISL.

Compétences

Maitrise de la signification et du comportement des langages.
Capacité d'analyse et de raisonnement sur ces objets.

Programme

Contenu

Cette UE permet d'aborder plusieurs méthodes de vérification de logiciels et d'utiliser leurs outils correspondants. Elle comporte 4 parties :

- **Correction d'un programme impératif**: Dans cette partie du cours, nous étudierons une méthode classique (la logique de Floyd-Hoare) permettant de prouver la correction d'un programme impératif par rapport à une spécification donnée. Nous présenterons tout d'abord brièvement les fondements théoriques de cette méthode, puis nous passerons à la pratique avec la plateforme Spark Ada. Cette plateforme, qui est utilisée dans l'industrie depuis de nombreuses années, permet de garantir la correction de systèmes critiques développés en Ada. Nous en profiterons pour rappeler les principes du paradigme "design-by-contract" adopté

par le standard 2012 du langage.

- **Utilisation de Tests pour la validation de logiciels**: Dans cette partie du cours, nous étudierons le développement de logiciels piloté par les tests en évaluant le coût des tests et le modèle de qualité FURPSE. Dans un deuxième temps nous étudierons les Tests dirigés par les modèles en détaillant l'utilisation des modèles dynamiques d'UML pour spécifier les tests.
- **Développement de programmes par la méthode B: de la spécification formelle à la génération automatique de code.** Dans cette partie du cours, nous étudierons à l'aide d'études de cas la méthode B pour spécifier formellement des systèmes. Nous vérifierons leur correction à l'aide du prouveur de l'Atelier B et enfin en utilisant des techniques de raffinement nous obtiendrons un code correct par construction.
- **Conception par utilisation de méthodes formelles: algèbre de processus** : Dans cette partie du cours, nous étudierons à l'aide d'études de cas l'utilisation d'une algèbre de processus pour spécifier formellement et vérifier des systèmes communicants. Nous utiliserons le langage LOTOS et la plateforme CADP.

Techniques et outils abordés dans les 4 parties de l'UE :

- Preuves de correction d'un programme impératif à l'aide de la logique de Floyd-Hoare. Utilisation de la plateforme Spark Ada.
- Développement de logiciels piloté par les tests et étude du modèle de qualité FURPSE. Étude des tests dirigés par les modèles à l'aide des modèles dynamiques d'UML.
- Développement de programmes corrects par construction à l'aide de la méthode B. Utilisation de l'Atelier B.

Spécification et vérification formelles de systèmes communicants à l'aide d'une algèbre de processus. Utilisation du langage LOTOS et de la plateforme CADP.

Bibliographie

Titre	Auteur(s)
Pratique des tests logiciels, Dunod, 2014.	Jacques Printz, Jean-Francois Pradat-Peyre
Model-based Testing: Where Does It Stand ?. CACM, Communications of the ACM, 58(2), 2015	Robert V. Binder, Bruno Legeard, and Anne Kramer
The B-Book: Assigning Programs to Meanings, Cambridge University Press, 2005.	J. R. Abrial