Modélisation et spécification formelles de logiciels

Coordonnateur du cours

VILLEMAIRE, Roger
villemaire.roger@uqam.ca
(514) 987-3000 #6744
PK-4615

Groupes

20	TERRASA, Alexandre	terrasa.alexandre@uqam.ca	PK-4470
	Mardi, de 9h30 à 12h30 Salle PK-2205 (cours)		
21	TERRASA, Alexandre	terrasa.alexandre@uqam.ca	PK-4470
	Mardi, de 18h00 à 21h00 Salle SH-3560 (cours)		

Description du cours

Le cours vise à initier les étudiants aux méthodes formelles de spécification et à leur rôle dans le cycle de développement des logiciels. Entres autres, il vise à familiariser les étudiants avec le mode descriptif de spécifications plutôt qu'avec le mode opérationnel (algorithme) auquel ils sont habitués. Il vise aussi à familiariser les étudiants avec les notions d'assertions, de contraintes et de contrats avec leur utilisation pour le développement de logiciels.

Rôle des spécifications et méthodes formelles.
Introduction à certaines notions formelles pour décrire des systèmes et composantes logiciels : diagrammes de classes et contraintes, automates et systèmes de transitions, contrats.
Approfondissement d'une approche basée sur les contraintes et contrats; logique : propositions et prédicats, quantificateurs, modélisation conceptuelle et description de propriétés; types abstraits : ensemble et multi-ensembles, séquences, spécifications comportementales, modélisation de composants divers (fonctions, types muables, types immuables), invariants, pré/post conditions.
Utilisation des assertions et contrats à l'étape de construction de logiciels : test unitaires, vérification dynamique de contrats.
Préalables académiques :
INF1130 Mathématiques pour informaticien ou MAT1060 Mathématiques algorithmiques ; INF2120 Programmation II

Objectifs du cours

Le cours vise à initier les étudiant-e-s aux méthodes formelles de spécification et à leur rôle dans le développement des logiciels. Entres autres, il vise à familiariser les étudiant-e-s avec le mode descriptif de spécification plutôt qu'avec le mode opérationnel auquel ils-elles sont habitué-e-s. Il vise aussi à familiariser les étudiant-e-s avec divers mécanismes d'abstraction utiles pour la description de composants et systèmes informatiques.

À la fin du cours, l'étudiant-e devrait être capable :

- de lire et naviguer dans des diagrammes de classe UML;
- d'expliquer les différents rôles que peuvent jouer les assertions dans le développement de logiciels;
- d'utiliser de façon pratique la logique et les assertions pour spécifier (c'est-à-dire, décrire et tester) le comportement et les propriétés d'opérations et de types de données;
- de manipuler des types abstraits de base (ensembles, séquences, sacs) et de les utiliser pour modéliser des objets ou types de données plus complexes;
- d'utiliser des expressions régulières et de les représenter par des automates;
- d'écrire des contraintes et contrats en utilisant la notation OCL;
- de lire et comprendre des contrats écrits dans d'autres langages (par ex., JASS, iContract, Eiffel ou JML);
- d'expliquer comment des assertions peuvent être utilisées pour obtenir des programmes plus robustes et faciliter le débogage.

Contenu du cours

- Introduction
  - Que sont les méthodes formelles ?
  - Aperçu d'OCL et de ses liens avec UML
- Logique (OCL)
  - Opérateurs et expressions logiques
  - Prédicats et quantificateurs
- Collections (OCL)
  - Types abstraits fondamentaux: Set, Bag, Sequence
  - Opérations produisant des collections (map)
  - Opérations produisant des valeurs (reduce)
  - Collections OCL vs. Collections Java (java.util.*); Valeurs vs. objets
- Modélisation conceptuelle UML
  - Diagrammes de classes
  - Navigation dans les classes et relations
  - Spécification de contraintes graphiques ou informelles (annotations)
- Expressions régulières et automates
  - Expressions régulières (Unix, Java)
  - Automates finis acceptants
  - Modèles conceptuels des automates (OCL)
- Tests unitaires et assertions
  - Évaluation dynamique des assertions: l'instruction assert
  - Rôle des tests
  - Exécution automatique des tests et assertions
  - Tests unitaires avec JUnit 4.0 (Java)
- Mise en oeuvre Java (5.0) des collections OCL
  - Types génériques
  - Itérateurs et boucle for
  - Interfaces, classes abstraites et classes concrètes
  - Méthodes de fabrication: Méthodes statiques, méthodes avec nombre variable d'arguments
  - Quantificateurs: Interfaces et classes internes anonymes
- Contrats (OCL)
  - Requêtes vs. commandes
  - Types abstraits: Collection de valeurs vs. classe d'objets;
- Débogage, contrats et assertions
  - Assertions en C
  - Utilisations diverses des assertions
- Contrats (suite) (non OCL)
  - Style "Modélisation abstraite" vs. style "Effets sur observateurs primitifs"
  - Exemples dans des langages divers: JASS, iContract, JML, Eiffel

Modalités d'évaluation

Description sommaire	Date	Pondération
Examen intra	Samedi, le 4 mars 9h30-12h30	35%
Examen final	Samedi, le 29 avril 9h30-12h30	35%

Deux (2) travaux pratiques		30%

L'utilisation de documentation personnelle est permise aux examens.
Une moyenne d'au moins 50% aux examens est nécessaire pour réussir le cours, mais non suffisante (il faut aussi réussir les travaux pratiques).
Les travaux pratiques doivent être réalisés préférablement en équipe de deux (2).
Aucun retard accepté dans la remise des TPs, sauf en cas d'entente préalable avec l'enseignant et uniquement pour des raisons dûment motivées (maladie grave, décès d'un proche, etc.).
La qualité du français sera prise en considération, tant dans les examens que dans les travaux pratiques (jusqu'à 10% de pénalité).
Les règlements concernant le plagiat seront strictement appliqués. Pour plus de renseignements, consultez le site suivant : http://www.sciences.uqam.ca/etudiants/integrite-academique.html

Politique d'absence aux examens

L'autorisation de reprendre un examen en cas d'absence est de caractère exceptionnel. Pour obtenir un tel privilège, l'étudiant-e doit avoir des motifs sérieux et bien justifiés.

Il est de la responsabilité de l'étudiant-e de ne pas s'inscrire à des cours qui sont en conflit d'horaire, tant en ce qui concerne les séances de cours ou d'exercices que les examens. **De tels conflits d'horaire ne constituent pas un motif justifiant une demande d'examen de reprise.**

Dans le cas d'une absence pour raison médicale, l'étudiant-e doit joindre un certificat médical original et signé par le médecin décrivant la raison de l'absence à l'examen. Les dates d'invalidité doivent être clairement indiquées sur le certificat. Une vérification de la validité du certificat pourrait être faite. Dans le cas d'une absence pour une raison non médicale, l'étudiant-e doit fournir les documents originaux expliquant et justifiant l'absence à l'examen – par exemple, lettre de la Cour en cas de participation à un jury, copie du certificat de décès en cas de décès d'un proche, etc. Toute demande incomplète sera refusée. Si la direction du programme d'études de l'étudiant-e constate qu'un étudiant a un comportement récurrent d'absence aux examens, l'étudiant-e peut se voir refuser une reprise d'examen.

L'étudiant-e absent-e lors d'un examen doit, dans les cinq (5) jours ouvrables suivant la date de l'examen, présenter une demande de reprise en utilisant le formulaire prévu, disponible sur le site Web du département à l'adresse suivante : **http://info.uqam.ca/politiques/**

L'étudiant-e doit déposer le formulaire dûment complété au secrétariat de la direction de son programme d'études : PK-3150 pour les programmes de premier cycle, PK-4150 pour les programmes de cycles supérieurs. Pour plus de détails sur la politique d'absence aux examens du Département d'informatique, consultez le site web suivant : **http://info.uqam.ca/politiques**

Intégrité académique

PLAGIAT Règlement no 18 sur les infractions de nature académique. (extraits)

Tout acte de plagiat, fraude, copiage, tricherie ou falsification de document commis par une étudiante, un étudiant, de même que toute participation à ces actes ou tentative de les commettre, à l'occasion d'un examen ou d'un travail faisant l'objet d'une évaluation ou dans toute autre circonstance, constituent une infraction au sens de ce règlement.

La liste non limitative des infractions est définie comme suit :

- **la substitution de personnes;**
- **l'utilisation totale ou partielle du texte d'autrui en la faisant passer pour sien ou sans indication de référence;**
- **la transmission d'un travail pour fins d'évaluatiion alors qu'il constitue essentiellement un travail qui a déjà été transmis pour fins d'évaluation académique à l'Université ou dans une autre institution d'enseignement, sauf avec l'accord préalable de l'enseignante, l'enseignant;**
- **l'obtention par vol, manoeuvre ou corruption de questions ou de réponses d'examen ou de tout autre document ou matériel non autorisés, ou encore d'une évaluation non méritée;**
- **la possession ou l'utilisation, avant ou pendant un examen, de tout document non autorisé;**
- **l'utilisation pendant un examen de la copie d'examen d'une autre personne;**
- **l'obtention de toute aide non autorisée, qu'elle soit collective ou individuelle;**
- **la falsification d'un document, notamment d'un document transmis par l'Université ou d'un document de l'Université transmis ou non à une tierce personne, quelles que aoient les circonstances;**
- **la falsification de données de recherche dans un travail, notamment une thèse, un mémoire, un mémoire-création, un rapport**

de stage ou un rapport de recherche;
- **Les sanctions reliées à ces infrations sont précisées à l'article 3 du Règlement no 18.**

Les règlements concernant le plagiat seront strictement appliqués. Pour plus de renseignements, veuillez consulter les sites suivants : **http://www.sciences.uqam.ca/etudiants/integrite-academique.html** et **http://www.bibliotheques.uqam.ca/recherche/plagiat/index.html**

Médiagraphie

VR MILES, R. and HAMILTON, K. -- *Learning UML 2.0* -- **O'Reilly Media, 2006.** Disponible dans Safari Books Online (accessible par la bibliothèque de l'UQAM http://www.computer.org/ ou de l'ACM http://www.acm.org/)

VR PILONE, D. and PITMAN, N. -- *UML 2.0 in a Nutshell* -- **O'Reilly Media, 2005** Disponible dans Safari Books Online (accessible par la bibliothèque de l'UQAM http://www.computer.org/ ou de l'ACM http://www.acm.org/)

UO www.info2.uqam.ca/~villemaire_r/3143.html
(énoncés des travaux pratiques, exemples, etc.)

UO www.info2.uqam.ca/~tremblay/INF3140
(notes de cours, exemples, anciens examens, etc.)

VR WARMER, J. and KLEPPE, A. -- *The Object Constraint Language - Second Edition : Getting Your Models Ready for MDA* -- **Addison-Wesley, 2003.**

VC BECK, K and GAMMA, E. -- *Test infected: Programmers love writing tests. Java Report, 3(7):37–50* -- **1998.**

AC BOWEN, J. and HINCHEY, M.G. -- *Seven more myths of formal methods* -- **IEEE Software, 12(4):34-41, July 1995.**

AC BOWEN, J. and HINCHEY, M.G. -- *Ten commandments of formal methods.* -- **IEEE Computer, 28(4):56-63, April 1995.**

AC BURDY, L., CHEON, Y., COK, D., ERNST, M., KINIRY, J., LEAVENS, G., LEINO, K., and POLL, E. -- *An overview of JML tools and applications. Int. J. Softw. Tools Technol. Transf., 7(3):212–232* -- **2005.**

AC HALL, A. -- *Seven myths of formal methods* -- **IEEE Software, 7(5):11-19, Sept. 1990.**

VC HUNT, A. and THOMAS D. -- *Pragmatic Unit Testing In Java with JUnit.* -- **The Pragmatic Bookshelf, Raleigh, NC, 2003.**

VC JACKSON, M. -- *Software Requirements & Specifications -- a lexicon of practice, principles and prejudices* -- **ACM Press & Addison-Wesley, 1995.**

AC LEAVENS, G. BAKER, A. and RUBY, C. -- *Preliminary design of JML: a behavioral interface specification language for Java. SIGSOFT Softw. Eng. Notes, 31(3):1–38* -- **2006.**

VC LISKOV, B. and GUTTAG, J. -- *Abstraction and specification in program development* -- **MIT Press, 1986.**

VC MESZAROS, G. -- *xUnit Test Patterns—Refactoring Test Code.* -- **Addison-Wesley, Upper Saddle River, NJ, 2007.**

VC MITCHELL, R. and McKIM, J. -- *Design by Contract, by Example* -- **Addison-Wesley, 2002**

VC TREMBLAY, G. -- *Modélisation et spécification formelle des logiciels (édition revue et augmentée)* -- **Loze-Dion Editeurs Inc., Montréal, 4e trimestre 2004.**

VC ZELLER, A. -- *Why Programs Fail -- A Guide to Systematic Debugging.* -- **Morgan Kaufmann Publishers and dpunkt.verlag, San Francisco, USA, 2006.**

A : article - C : comptes rendus - L : logiciel
S: Standard - U : uri - V : volume
C : complémentaire - O : Obligatoire - R : recommandé