INF3140 – Modélisation et spécification formelles de logiciels

Plan de cours – Automne 2009

COORDONNATEUR		TREMBLAY, Guy	tremblay.guy@uqam.ca	(514) 987-3000 8213	PK-4435
GROUPE	10	TREMBLAY, Guy	tremblay.guy@uqam.ca	(514) 987-3000 8213	PK-4435
		Lundi, de 13h30 à 15h00 et jeudi, de 15h30 à 17h00			

DESCRIPTION

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 spécification plutôt qu'avec le mode opérationnel (algorithmique) auquel ils sont habitués. Il vise aussi à familiariser les étudiants avec divers mécanismes d'abstraction utiles pour la description de composants et systèmes informatiques. Rôle des spécifications et méthodes formelles. Introduction à certaines notations formelles pour décrire les exigences et spécifications de composants et systèmes logiciels: modélisation abstraite, spécifications algébriques des types abstraits et/ou automates et systèmes de transition. Approfondissement d'une méthode basée sur la modélisation abstraite – logique: propositions et prédicats, quantificateurs, description du domaine d'application et descriptions de propriétés; types abstraits: ensembles, séquences, dictionnaires; spécification de systèmes et composants logiciels: spécification comportementale abstraite, modélisation de diverses sortes de modules (machine vs. classe vs. type immuable), invariant, pré/post-conditions, exceptions; méthode rigoureuse de développement: analyse des propriétés, biais d'implantation, raffinement et mise en oeuvre.

Préalables: INF1130 Mathématiques pour informaticien ou MAT2055 Logique et ensembles ; INF2120 Programmation II

OBJECTIFS

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.

ÉVALUATION

Description sommaire	Date	Pondération
Examen intra	Lundi 26 octobre 2009 – 13h30 à 16h30	30%
Examen final	Lundi 21 décembre 2009 – 13h30 à 16h30	35%
Trois (3) travaux pratiques		35%

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).

Une pénalité de 10% par jour de retard sera appliquée.

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, veuillez consulter les sites suivants :

http://www.sciences.uqam.ca/decanat/reglements.php

http://www.bibliotheques.uqam.ca/recherche/plagiat/index.html

Politique d'absence aux examens

Un étudiant absent à un examen se verra normalement attribuer la note zéro pour cet examen. Cependant, si l'étudiant était dans l'impossibilité de se présenter à l'examen pour un motif valable, certains arrangements pourront être pris avec son enseignant. Pour ce faire, l'étudiant devra présenter à son enseignant l'un des formulaires prévus à cet effet accompagné des pièces justificatives appropriées (par ex., attestation d'un médecin que l'étudiant était dans l'impossibilité de se présenter à l'examen pour des raisons de santé, lettre de la Cour en cas de participation à un jury).

Une absence pour cause de conflit d'horaires d'examen n'est pas considérée comme un motif valable d'absence, à moins d'entente préalable avec la direction du programme et l'enseignant durant la période d'annulation des inscriptions avec remboursement : tel qu'indiqué dans le guide d'inscription des étudiants, il est de la responsabilité d'un étudiant de ne s'inscrire qu'à des cours qui ne sont pas en conflit d'horaire.

Pour plus de détails sur la politique d'absence aux examens du Département d'informatique et pour obtenir les formulaires appropriés, consultez le site web suivant :
http://www.info.uqam.ca/enseignement/politiques/absence-examen

CONTENU

- 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

RÉFÉRENCES

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

U O **http://www.info2.uqam.ca/~tremblay/INF3140**
Ce site contient, entre autres, des transparents, des énoncés d'anciens examens, des exercices (énoncés, solutions), etc.

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

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

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

A C 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.**

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

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

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

A C 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.**

V C LISKOV, B. and GUTTAG, J. – *Abstraction and specification in program development* – **MIT Press, 1986.**

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

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

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

V C 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 – N : notes – R : revue – S : standard – U : uri – V : volume

C : complémentaire – O : obligatoire – R : recommandé