

COORDONNATEUR	FRIEDMANN, Alex	friedmann.alexander@uqam.ca	(514) 987-3000 3219	PK-4530
GROUPE	10 TREMBLAY, Guy	tremblay.guy@uqam.ca	(514) 987-3000 8213	PK-4435
	Lundi, de 13h30 à 15h00 et mercredi, de 15h00 à 16h30			

DESCRIPTION	<p>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.</p> <p>Préalables: INF1130 Mathématiques pour informaticien ou MAT2055 Logique et ensembles ; INF2120 Programmation II</p>
-------------	---

OBJECTIFS	<p>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.</p> <p>À la fin du cours, l'étudiant-e devrait être capable :</p> <ul style="list-style-type: none"> d'expliquer le rôle des méthodes formelles dans le cycle de vie des logiciels et d'identifier les avantages et limites de leur utilisation; d'utiliser de façon pratique la logique et des assertions pour spécifier (c'est-à-dire, décrire et tester) les propriétés d'opérations et types de données; de manipuler des types abstraits de base (ensembles, séquences, dictionnaires) et de les utiliser pour modéliser des objets ou types de données plus complexes; d'écrire des spécifications comportementales formelles en utilisant la notation Spec; de lire et comprendre des spécifications formelles écrites en OCL 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 20 octobre 13h30-16h30	30%
	Examen final	Lundi 15 décembre 13h30-16h30	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 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).

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: Qu'est-ce que les méthodes formelles?
 - Que sont les méthodes formelles?
 - Quels sont leurs bénéfices?
 - Quand doit-on les utiliser?
- ❑ Logique et concepts
 - Propositions et prédicats
 - Quantificateurs et formes génératrices
 - Concepts et modules de définitions: désignations vs. définitions
 - Généricité et héritage
- ❑ Types abstraits Spec
 - Types de base: boolean, nat, integer, real, char, string
 - Ensembles: valeurs, relations et opérations; techniques implicites vs. explicites de spécifications
 - Séquences: valeurs, constructeurs et opérations;
 - Dictionnaires: valeurs, constructeurs et opérations; techniques implicites de spécifications
 - Autres types: énumérations, tuples
 - Comparaison avec les collections Java
- ❑ Assertions et tests unitaires
 - Tests unitaires
 - Cadres de tests XUnit et assertions
 - JUnit 4.0
- ❑ Spécifications fonctionnelles à l'aide de pré/post-conditions (en Spec)
 - Rôle des spécifications fonctionnelles
 - Les diverses formes de modules Spec: fonctions, machines, types
 - Exemples de modules FUNCTION
 - Spécifications complètes vs. partielles
 - Spécification des réponses et des exceptions (responsabilité du client vs. du serveur)
- ❑ Machines abstraites (en Spec)
 - Rôle des machines
 - Exemples: systèmes de gestion d'information, contrôleurs de machines à états finis
 - Convention pour la description des transitions
 - Diagrammes de flux des messages et place de l'interface personne-machine
 - Formes générales des messages et réponses
 - Diverses formes d'envoi de message (explicite vs. implicite, synchrone vs. asynchrone)
- ❑ Types mutables et immuables (en Spec)
 - Similitudes et différences entre types mutables et immuables: Classes d'objets vs. collections de valeurs
 - Exemples de types mutables: types de données, boîtes vocales; mise en oeuvre dans un langage de programmation
 - Exemples de types immuables: types de données, chaînes de caractères; mise en oeuvre dans un langage de programmation

- ❑ Conception par contrat et programmation avec assertions
 - Contrat et assertions
 - Support des assertions dans divers langages
 - Heuristiques pour l'utilisation des assertions: pré-conditions, post-conditions, explicitation d'hypothèses, tests de conditions impossibles; danger des effets de bord
 - Rôle des assertions pour le débogage
- ❑ La notation OCL d'UML et/ou la notation JML (selon le temps disponible)
 - Éléments de base de la notation
 - Correspondance avec les expressions Spec
 - Description formelle de types mutables Java

RÉFÉRENCES

- VO TREMBLAY, G. – *Modélisation et spécification formelle des logiciels (édition revue et augmentée)* – Loze-Dion Editeurs Inc., Montréal, 4e trimestre 2004.
- UC <http://www.info2.uqam.ca/~tremblay/INF3140>
Ce site contient, entre autres, des transparents, des énoncés d'anciens examens, des exercices (énoncés, indices, solutions), etc.
- VC K. Beck and E. Gamma. – *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 Berzins, V. and Luqi – *An introduction to the specification language Spec* – *IEEE Software*, 7(2):74-84, March 1990.
- VC Berzins V. and Luqi – *Software Engineering with Abstractions* – Addison-Wesley Publishing Co., 1991.
[En réserve: QA76.76D47B47]. Le livre de référence pour le langage Spec. Couvre l'ensemble du cycle de vie. Définition détaillée de la syntaxe de Spec et spécification complète de la bibliothèque des types Spec.
- AC L. Burdy, Y. Cheon, D. Cok, M. Ernst, J. Kiniry, G. Leavens, K. Leino, and E. Poll. – *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 A. Hunt and D. Thomas. – *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 G. Leavens, A. Baker, and C. Ruby. – *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 G. Meszaros. – *xUnit Test Patterns—Refactoring Test Code*. – Addison-Wesley, Upper Saddle River, NJ, 2007.
- VC J. Warmer and A. Kleppe. – *The Object Constraint Language—Precise Modeling with UML*. – Addison-Wesley, 1999.
- VC A. Zeller. – *Why Programs Fail — A Guide to Systematic Debugging*. – Morgan Kaufmann Publishers and dpunkt.verlag, San Francisco, USA, 2006.
- LO *Analyseur Spec* – <http://www.info2.uqam.ca/~inf3140/analyseur-spec.cgi>

A : article – C : comptes rendus – L : logiciel – N : notes – R : revue –
S : standard – U : uri – V : volume

C : complémentaire – O : obligatoire – R : recommandé