PLAN DE COURS
Été 2021

UQAC

8INF958 • Spécification, test et vérification

Enseignant:	Sylvain Hallé
Bureau:	P4-5240
Courriel:	shalle@uqac.ca
Page web du cours:	https://moodle.uqac.ca
Disponibilité:	Sur rendez-vous

Contenu général

On rapporte que chez Microsoft, près d'un programmeur sur deux est chargé de tester les produits, et non de les développer. Cette donnée surprenante est une conséquence du fait que la croissance des capacités des ordinateurs s'est accompagnée d'une croissance encore plus grande de la complexité de leurs programmes —et de la difficulté à prévoir toutes leurs exécutions possibles et à retracer toutes les erreurs imaginables. Le développement d'applications informatiques s'appuie donc largement sur la notion de test : on imagine des séquences d'actions ou d'entrées à soumettre au programme visant à révéler des potentielles erreurs de programmation. Cette technique, qui ne fournit aucune garantie formelle de qualité, commence à montrer ses limites à mesure que les systèmes informatiques gagnent en complexité.

Ce cours se veut une introduction aux concepts fondamentaux de la spécification, du test et de la vérification automatiques de systèmes informatiques. On y verra comment spécifier le fonctionnement attendu d'un système au moyen de langages formels comme les automates, la logique propositionnelle et la logique temporelle. On étudiera ensuite les méthodes permettant de vérifier, par différents moyens, qu'un système donné satisfait bien la spécification fournie. Finalement, les concepts théoriques seront accompagnés d'une variété d'exercices pratiques, qui seront l'occasion d'utiliser plusieurs logiciels de test et de vérification automatique.

Objectifs du cours

Au terme de ce cours, l'étudiant aura acquis les compétences suivantes :

- Modéliser des systèmes au moyen de différents langages de spécification
- Comprendre et appliquer des techniques de génération et d'automatisation des tests
- Identifier les limites de l'approche par tests dans le développement de logiciels
- Comprendre et appliquer des algorithmes de vérification automatique (satisfaisabilité, runtime monitoring, model checking)
- Utiliser une variété d'outils automatiques pour vérifier qu'un système satisfait une spécification

Sujets abordés

Un aperçu du contenu de chaque leçon est donné ci-dessous. L'ordre de présentation peut être modifié selon les besoins. Chaque leçon ne correspond pas nécessairement à une séance exacte.

Bloc A : tests et qualité

Dans ce premier bloc, on s'intéressera aux concepts généraux entouant les bugs informatiques, la qualité et les tests logiciels.

1. Généralités Les bugs informatiques et leurs conséquences. Que sont les tests, et pourquoi tester un logiciel ? Aperçu des travaux du LIF.

2. Concepts de base Définition du concept de test. Les catégories de tests. Les objectifs d'une suite de tests. Les sept principes du test logiciel.

3. Automatisation et intégration continue Introduction à l'intégration continue. L'analyse statique de code : règles de style et *code smells*. Utilisation d'une librairie d'automatisation de tests unitaires.

Bloc B : énoncer des conditions

Dans ce deuxième bloc, on verra comment utiliser différentes formes de notations pour décrire le comportement attendu d'un système de diverses façons.

4. Conditions booléennes Introduction à la logique propositionnelle : syntaxe et sémantique. Notions de valuations, satisfaisbilité. Utilisation d'un solveur SAT pour résoudre des contraintes.

5. Quantificateurs Logique du premier ordre : notions de quantificateurs, interprétations, indécidabilité. Application à la spécification au moyen du langage OCL.

6. Séquences d'exécution Automates et diagrammes d'états UML, logique temporelle linéaire (LTL).

Bloc C : générer des tests

Le troisième bloc est consacré à diverses techniques pour générer des suites de tests pour un système donné.

7. Tests fonctionnels I Introduction aux tests fonctionnels. Analyse des valeurs limites, partitionnement en classes d'équivalence.

8. Tests fonctionnels II Tests combinatoires. Tests basés sur les propriétés (*property-based testing*). Utilisation d'un outil de génération de tests combinatoires.

9. Tests structurels Diverses métriques de couverture d'une suite de tests. Exécution symbolique.

10. Sujets avancés Un aperçu de techniques de tests moins connues ou plus expérimentales. On pourra y aborder entre autres : tests de mutation, delta-debugging.

Bloc D : méthodes formelles

Le dernier bloc s'intéresse aux méthodes alternatives aux tests traditionnels afin de détecter des défauts dans l'implémentation d'un système.

11. Surveillance à l'exécution Instrumenter un programme : librairies de journalisation (*logging*), notions de programmation orientée par aspects. Le *runtime monitoring* ; utilisation d'un logiciel

de monitoring pour détecter des bugs. Au-delà du monitoring : le Complex Event Processing. Préssentation du logiciel BeepBeep 3.

12. Approches statiques I Le model checking : la logique CTL et son algorithme de vérification. Modélisation et vérification d'un problème au moyen d'un model checker.

13. Approches statiques II Programmation par contrats. Annotations d'un programme avec le langage JML. Vérification statique d'annotations et preuves de programmes.

Évaluation

L'évaluation consistera en les éléments suivants :

- Quatre devoirs (un par bloc de contenu), à effectuer en équipes d'au plus trois personnes. Les devoirs consistent en la réalisation de travaux pratiques nécessitant pour la plupart l'écriture de code. Les dates de remise de ces devoirs sont respectivement les **23 mai**, **6 juin**, **20 juin** et **4 juillet**. Le total des devoirs comptera pour **50 points** dans la note finale, avec une pondération égale pour chacun.
- Un travail s'échelonnant sur l'ensemble de la session, portant sur l'étude d'un outil existant ou d'un article scientifique relatif au contenu du cours. L'évaluation du travail sera effectuée par la production d'un rapport écrit de 20 pages, à remettre au plus tard le **6 juillet**. Ce travail comptera pour **25 points**.
- Un examen final à choix multiples, à réaliser en ligne sur la plateforme Moodle, le **5 juillet**. Cet examen compte pour **25 points**.

La note finale, sur 100, est la somme de toutes ces évaluations. La note de passage est fixée à **50%** ou C. (L'attribution des notes C–, D+ et D s'applique uniquement aux cours de premier cycle.) Une pénalité de 5% par jour sera appliquée aux travaux remis en retard ; un travail remis avec plus de trois jours de retard ne sera pas corrigé.

Formule pédagogique

Encore une fois pour la session d'été 2021, le cours sera dispensé exclusivement en ligne de manière asynchrone. Toutes les interactions se feront à travers la page du cours sur la plateforme Moodle de l'UQAC (voir instructions plus bas).

À chaque semaine, une ou deux nouvelles leçons seront dévoilées sur la plateforme Moodle. Chacune de ces leçons sera structurée de la même manière, et contiendra :

- Une capsule vidéo où l'enseignant donne un aperçu de la matière de cette leçon, présente le contenu et au besoin donne des explications sur les travaux ou les exercices qui lui sont associés.
- Une série de diapositives décrivant la matière. Ces diapositives seront accompagnées d'une narration.
- Des lectures tirées d'extraits des manuels donnés en référence. Toutes ces lectures seront mises à la disposition des étudiants sous la forme de documents PDF. La plupart d'entre elles sont en anglais.
- De courts exercices *facultatifs* destinés à approfondir la connaissance de la matière de la leçon. (La remise de ces exercices est facultative, et ceux-ci ne comptent pas dans la note

finale.)
- Pour certaines des leçons, l'énoncé d'un des quatre devoirs à remettre aux dates indiquées plus haut.

Les étudiants pourront interagir avec l'enseignant au moyen d'un forum situé sur la page Moodle du cours. Des rencontres individuelles via la plateforme Zoom pourront également être prévues sur demande.

Préalables techniques

Le cours nécessite de l'étudiant une connaissance élémentaire de la programmation orientée-objet en Java. Des ressources de rattrapage seront fournies en début de session.

Pour réaliser les exercices et les devoirs, le cours nécessite que l'étudiant ait accès à un ordinateur sur lequel sont installés :
- Un kit de développement Java (Java Development Kit ou JDK), version 8 ou supérieure. Il existe deux versions gratuites disponibles pour toutes les plateformes (Mac, Linux, Windows) : le Sun JDK [1] et le OpenJDK [2].
- L'environnement de développement Eclipse [3], lui aussi disponible gratuitement pour toutes les plateformes. Il existe d'autres IDE pour Java, mais les exercices et les devoirs nécessitent l'utilisation de plug-ins qui sont spécifiques à Eclipse.

Utilisation des TIC

Ne s'applique pas, étant donné qu'aucune séance en classe n'aura lieu.

Qualité du français écrit

Tout travail remis doit être conforme aux exigences de la politique institutionnelle en matière de maitrise du français écrit. Comme il en est fait mention dans le Manuel de gestion (3.1.1-012), [4] tout travail dont la qualité du français serait jugée non conforme par l'enseignant pourra être pénalisé jusqu'à concurrence de 10% du résultat maximal prévu.

Évaluation de la qualité de l'enseignement

Ce cours sera évalué en fonction de la Procédure relative à l'évaluation des activités aux programmes d'études de cycles supérieurs (Manuel de gestion, 3.1.2-008).

Situation du cours dans le programme

Le cours est optionnel pour les étudiants inscrits aux programmes de cycles supérieurs. Aucun cours ne lui est préalable.

1. https://www.oracle.com/java/technologies/javase-downloads.html
2. https://openjdk.java.net/
3. https://eclipse.org
4. http://www.uqac.ca/direction_services/secretariat_general/manuel/index.pdf

Références

Aucun livre ou recueil de notes n'est obligatoire pour ce cours. Les lectures et les diapositives utilisées durant les séances seront rendues disponibles en format électronique sur le site (Moodle) du cours. Les lectures sont tirées des ouvrages suivants, qui sont suggérés comme de bons points de départ pour la matière vue dans ce cours.

- P. Amman, J. Offutt. (2017). Introduction to Software Testing. Cambridge University Press. ISBN 978-1107172012.
- R. Black, E. van Veenendaal, D. Graham. (2015). Foundations of Software Testing — ISTQB Certification. Cengage Learning EMEA, ISBN 978-8131526361.
- E.M. Clarke, O. Grumberg, D.A. Peled. (2000). Model Checking. MIT Press, ISBN 0-262-03270-8.
- P.M. Duvall, S. Matyas, A. Glover. (2007). Continuous Integration : Improving Software Quality and Reducing Risk. Addison Wesley. ISBN 978-0321336385.
- M. Fowler. (2003). UML Distilled : A Brief Guide to the Standard Object Modeling Language. Addison Wesley. ISBN 978-0321193681.
- M. Huth, M. Ryan. (2004). Logic in Computer Science : Modelling and Reasoning about Systems. Cambridge University Press, ISBN 978-0521543101.
- D.R. Kuhn, R.N. Kacker, Y. Lei. (2013). Introduction to Combinatorial Testing. Chapman and Hall/CRC. ISBN 978-1466552302.
- S. McConnell. (2017). Code Complete, 2nd Edition. Microsoft Press. ISBN 0-7356-1967-0.
- R. Miles. (2004). AspectJ Cookbook. O'Reilly, 978-0596006549.
- G.J. Myers. (2004). The Art of Software Testing, 2nd Edition. Wiley. ISBN 0-471-46912-2.
- A. Page, K. Johnston, Bj Rollinson. (2008). How We Test Software at Microsoft. Microsoft Press, ISBN 978-0735624252.

Site du cours

Toutes les ressources en ligne relatives au cours se trouvent sur le site Moodle de l'université, à l'adresse `http://moodle.uqac.ca`.

Identifiant : _____

Mot de passe : _____

Si un compte Moodle n'a pas déjà été créé, il est de la responsabilité de l'étudiant de se faire ajouter au groupe « 8INF958 (Sylvain Hallé) », groupe 01, en contactant le support informatique à l'adresse `supportsti@uqac.ca` ou par téléphone au 418-545-5011, poste 6000.