UNIVERSITÉ DE SHERBROOKE

Faculté des sciences
Centre de formation en technologies de l'information

Cours :	INF 752 – Techniques de vérification et de validation
Trimestre :	Automne 2018
Enseignant :	Évariste Valéry BÉVO WANDJI

1. Mise en contexte

La vérification des systèmes parallèles ou répartis/distribués demeure encore aujourd'hui un défi majeur pour l'industrie du développement logiciel. Or de tels systèmes, constituées de parties qui fonctionnent concurremment, mais avec des liens de communication entre elles, sont monnaie courante aujourd'hui (systèmes de télécommunication, applications Internet, jeux, les logiciels de contrôle de robots mobiles et autres).

La conception et la réalisation de tels systèmes donnent lieu à des types d'anomalies qu'on ne rencontre pas dans des systèmes séquentiels/centralisés conventionnels et qui seraient difficiles à détecter sans outil approprié (interblocage, famine, vivacité et autres). Dans ce contexte, plusieurs de travaux de recherche et développement ont été et sont encore consacrés au développement de techniques et d'outils permettant la vérification et validation de tels systèmes (garantir qu'ils fonctionneraient correctement quels que soient les scénarios envisagés lors de l'exécution).

Dans ce cours, il est question d'apprivoiser une de ces techniques de vérification de systèmes parallèles ou répartis/distribués : la « vérification de modèles » (ou « model-checking » en anglais). Avec cette technique, pour vérifier un système, on le modélise d'abord à l'aide de transitions dans un espace d'états (c'est-à-dire les états possibles du système). Ensuite, on modélise les propriétés du système qu'on veut vérifier (par exemple, le fait que toute perte de message sera détectée ou la propriété que les processus parallèles du système ne se bloquent pas mutuellement et indéfiniment). Ensuite, on utilise un logiciel de vérification qui prend en entrée le modèle du système et le modèle des propriétés et donne en sortie des traces d'exécution du modèle qui violent la propriété, s'il y en a. S'il n'en y a pas, c'est que la conception du système (via le model) est correcte.

La « vérification de modèles » étant une technique de vérification/validation parmi tant d'autres, la première partie du cours est consacrée à une sorte de mise en contexte, avec un survol des principales techniques de vérification et validation en génie logiciel, et un examen un peu plus détaillé de quelques techniques classiques de vérification et validation (revues et tests).

Le cours se veut à la fois théorique est pratique. Sur le plan théorique, la vérification de modèles est fondée sur des théories bien établies qui seront exposées dans ce cours. Sur le

plan pratique, des algorithmes basés sur ces théories ont été développés et intégrés dans des logiciels de vérification de domaine public ou commercial. Un de ces logiciels (Spin) sera utilisé dans le cadre d'un projet simple.

2. Place du cours dans le programme

Qu'il s'agisse du programme de deuxième cycle en génie logiciel ou encore du programme de maîtrise en génie logiciel incluant un cheminement de type cours en technologies de l'information, le cours permettra à l'étudiant ou l'étudiante d'approfondir ses connaissances en matière de vérification/validation de systèmes logiciels en général, vérification de systèmes parallèles en particulier (utilisation de la technique « vérification de modèles » (ou « model-checking » en anglais)). Le cours est conçu de façon à intégrer des étudiants ou des étudiantes ayant différents niveaux de connaissances et d'expérience en matière de développement logiciel, en fournissant un cadre de référence uniforme pour l'étude des techniques de vérification et de validation. Il est toutefois supposé que les étudiantes ou les étudiants sont familiers avec le processus de développement logiciel.

3. Objectifs généraux

(Selon l'annuaire des activités pédagogiques, disponible sur le site de l'Université : http://www.usherbrooke.ca/programmes/cours/INF/inf752.htm)

Cette activité pédagogique vise à développer chez l'étudiante ou l'étudiant les aptitudes suivantes :

- identification et utilisation des techniques de vérification et de validation;
- modélisation d'un système logiciel ou d'un protocole à l'aide de transitions dans un espace d'états ;
- modélisation des propriétés d'un système logiciel qu'on veut vérifier ;
- utilisation d'un outil de vérification de modèles.

4. Objectifs spécifiques

À la fin de cette activité pédagogique, l'étudiante ou l'étudiant sera capable de :

- classer et comparer les techniques de vérification et de validation;
- planifier et réaliser des tests ou des revues formelles classiques;
- expliquer en grandes lignes en quoi consistent les problèmes de vérification de programmes et tests de conformité;
- analyser un texte scientifique portant sur les techniques de vérification et validation;
- modéliser des systèmes/protocoles en langage Promela et les simuler à l'aide SPIN;
- utiliser les automates sur mots infinis pour exprimer des propriétés de systèmes;

- écrire, lire et comprendre des formules LTL (une alternative pour la spécification des propriétés de systèmes).
- utiliser SPIN et Promela pour vérifier des propriétés de systèmes;

5. Planification hebdomadaire

Semaine	Contenu du cours	Travaux	Poids
1	Présentation du contenu du cours et entente d'évaluation Introduction à la vérification et validation des systèmes logiciels : • terminologie; • principes et techniques de vérification et validation.		
2	Vérification et validation : les tests	*Plan détaillé de test à produire (proposition d'une spécification)*	
3	Vérification et validation : les tests		
4	Vérification et validation : les revues		
5	Vérification et validation : les revues	*Plan de test (remise des travaux au prof.)*	15%
6	Langage Promela		
7	**Examen de mi-session**	*Examen de mi-session* *Projet de session (énoncé)*	25%
8	SPIN - le simulateur : • Simulation d'un programme Promela avec SPIN • Petite révision		
9	Logique propositionnelle & des prédicats : • Les bases de la logique propositionnelle & des prédicats • Spécifier avec des formules de la logique		
10	Automates et Promela (spécification de propriétés) : • Automate sur mots infinis • Modèle de Kripke • La sémantique de Promela		
11	Automates et Promela (spécification de propriétés) : • Vérification basique avec SPIN • Utilisation d'automates		

12	LTL (spécification de propriétés) : • Spécifier en LTL • Vérification avec SPIN		
13	LTL (spécification de propriétés) : • Spécifier en LTL • Vérification avec SPIN		
14	Récapitulatif du cours et conclusion		
15	**Examen final**	*Examen final* *Projet de session (remise des travaux au professeur)*	35% 25%

6. Approche pédagogique préconisée

L'approche pédagogique qui permettra d'atteindre les objectifs visés par le cours est la suivante : un enseignement hebdomadaire sous la forme d'un cours magistral, avec exemples et discussions en classe pour une période de trois heures. L'étudiant ou l'étudiante devra compléter sa formation par des travaux individuels proposés et des lectures personnelles suggérées. Le cours pourra être adapté selon les besoins des étudiants et étudiantes, si nécessaire.

7. Évaluation de l'apprentissage

Description sommaire	Pondération	Pondération individuelle	Pondération de groupe
Travail No1 (*travail individuel)* • *production d'un plan détaillé de test d'un module de système logiciel à partir de ses spécifications*	15%	☑	☐
Projet de session (*par équipe de 3 au maximum)* *Le projet consistera à :* *a) trouver un système parallèle ou distribué à vérifier (par exemple un protocole de communication)* *b) modéliser ce système en Promela;* *c) modéliser des propriétés du système en Promela et en LTL et les vérifier à l'aide de SPIN;*	25%	☐	☑
Examen de mi-session : • *travail individuel* • *questions de cours et exercices*	25%	☑	☐

Examen final : • *travail individuel* • *L'examen sera constitué de questions techniques (par exemple, écrire un petit programme Promela qui modélise un processus donné; décrire le comportement de processus Promela donné; ou donner une formule LTL pour une propriété quelconque) et/ou théorique (par exemple, des questions cherchant à vérifier si vous avez bien compris les différences fondamentales entre les différentes approches de V&V).*	35%	☑	☐
Notez bien : Pour tous les travaux, un rapport classique de TP (documents papier et électronique) doit être remis au professeur. La qualité du français compte pour 5% de la note. La pénalité pour le retard dans la remise d'un travail est de 5% de la note par jour ouvrable de retard.			

Toute modification reliée à une date de remise doit avoir été acceptée par le groupe et la direction du CeFTI dans un délai plus grand qu'une semaine avant l'échéance de la remise.

8. Plagiat

Un document dont le texte et la structure se rapportent à des textes intégraux tirés d'un livre, d'une publication scientifique ou même d'un site Internet, doit être référencé adéquatement. Lors de la correction de tout travail individuel ou de groupe une attention spéciale sera portée au plagiat, défini dans le Règlement des études comme "le fait, dans une activité pédagogique évaluée, de faire passer indûment pour siens des passages ou des idées tirés de l'œuvre d'autrui." Le cas échéant, le plagiat est un délit qui contrevient à l'article 8.1.2 du Règlement des études : "tout acte ou manœuvre visant à tromper quant au rendement scolaire ou quant à la réussite d'une exigence relative à une activité pédagogique.". À titre de sanction disciplinaire, les mesures suivantes peuvent être imposées: a) l'obligation de reprendre un travail, un examen ou une activité pédagogique et b) l'attribution de la note E ou de la note 0 pour un travail, un examen ou une activité évaluée. Tout travail suspecté de plagiat sera référé à la vice-doyenne à l'enseignement de la Faculté des sciences.

9. Adresse électronique pour remise des travaux

evariste.valery.bevo.wandji@usherbrooke.ca

10. Bibliographie

Manuel Obligatoire

[1] Moti Ben-Ari, *Principles of Spin Model Checker*, Springer Verlag, Jan. 2008

Autres références

[1] Claude Y. Laporte, Alain April - *L'assurance qualité logicielle 2 : processus de support,* LAVOISIER, Paris, 2011, ISBN 978-2-7462-3222-8.
[2] Claude Y. Laporte, Alain April - *L'assurance qualité logicielle 1 : concepts de base*, LAVOISIER, Paris, 2011, ISBN 978-2-7462-3147-4.
[3] Linda Westfall, *The Certified Software Quality Engineer Handbook*, Sep 1 2009
[3] Linda Westfall, *The Certified Software Quality Engineer Handbook*, Second Edition, Milwaukee, Wisconsin - ASQ Quality Press, ISBN: 978-0-87389-939-0, 2017
[4] Cristel Baier, Joost-Pieter Katoen, Kim Larsen, *Principles of Model Checking*, MIT Press, May 2008.
[5] Galin, D., *Quality Software Assurance*, Pearson Addison-Wesley, 2004
[6] **Gerard J. Holzmann, *The Spin Model Checker: Primer and Reference Manual*, Addison-Wesley, 2004.**
[7] Lisa Crispin, Janet Gregory, *Agile Testing: A Practical Guide for Testers and Agile Teams*, Dec 30 2008
[8] Sue Carroll and Taz Daughtrey, *Fundamental Concepts for the Software Quality Engineer, Volume 2*, Jun 1 2007
[9] Schulmeyer, G., *Verification & Validation of Modern Software Intensive Systems*, Prentice Hall, 2000
[10] Gerald D. Everett, Raymond McLeod, *Software testing: testing across the entire software development life cycle*, Wiley-IEEE, 2007
[11] Ronald A. RADICE, *High Quality Low Cost Software Inspections*, Paradoxicon Pub, 2002
[12] Karl Wiegers, *Peer Reviews in Software - A Practical Guide*, Addison-Wesley, 2001
[13] Dustin, Elfriede, *Effective Software Testing*. Addison Wesley, 2002.

Liens Web

[1] ISO/IEC 25010:2011, Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models - https://www.iso.org/standard/35733.html

[2] Test Maturity Model integration (TMMi) - https://www.tmmi.org/

[3] IEEE 1012-2012 IEEE Standard for System and Software Verification and Validation. 25 May 2012. ISBN 978-0-7381-7268-2. doi:10.1109/IEEESTD.2012.6204026 - http://ieeexplore.ieee.org/document/6204026

Logiciel SPIN

http://spinroot.com/

Logiciel•UPPAAL

Uppaal est un environnement intégré pour la vérification et la modélisation de système en temps réel tels que les réseaux d'automates temporels étendus avec des types de données. Cet outil a été développé en collaboration entre le groupe Design and Analysis of Real-

Time Systems de l'Université d'Uppsala en Suède et le groupe Basic Research in Computer Science à l'Université d'Aalborg au Danemark.
http://www.uppaal.com/

Logiciel CADP

CADP est un environnement pour la vérification de programmes. Il possède de nombreux atouts et est très flexible. Il est possible d'utiliser de nombreux mécanismes pour vérifier des propriétés : logique temporelle; bisimulation; etc. Il a été développé par l'INRIA Rhône-Alpes en France.
http://www.inrialpes.fr/vasy/cadp

Testing tools

http://www.softwareqatest.com/qattls1.html