Logiciels Sûrs ***Reliable Softwares***			
Code cours *Course code*: **LOS**		**Crédits ECTS** *ECTS Credits:* **2**	
Département *Department* **Coordonnateurs** *Lecturers*	: IA : Allel HADJALI	**Cours** *Lectures* **T.D.** *Tutorials* **T.P.** *Laboratory sessions*	: 10h : 10h :
Période *Year of study*	: 3ième année *3rd year*	**Projet** *Project* **Non encadré** *Homework*	: :
Semestre *Semester*	: 5ième semester – B *5th semester - B*	**Horaire global** *Total hours*	: 20h
Evaluation *Assessment method(s)*	: Examen écrit / written Exam		
Langue d'instruction *Language of instruction*	: Français / French		
Type de cours *Type of course*	*:* Obligatoire / Compulsory		
Niveau *Level of course*	: 2e cycle / Graduate		

Compétences attendues :

Maitriser le développement de logiciels sûrs, fiables et robustes. Ces logiciels occupent une place fondamentale, notamment dans les secteurs critiques comme le transport, l'avionique, le spatial et l'énergie. Pouvoir utiliser les méthodes formelles pour modéliser les spécifications, écrire les preuves, vérifier et prouver la correction des programmes et autres propriétés utiles en Génie Logiciel.

Pré-requis : Connaissances en logique, concepts fondamentaux de programmation, modélisation et algorithmique.

Contenu :

Ce cours présente les méthodes de vérification formelle de programmes classiques et également des logiciels dédiés aux systèmes réactifs et concurrents. Après une brève introduction sur les qualités attendues des logiciels, notamment, dans les domaines critiques, une présentation des outils mathématiques nécessaire à la vérification formelle est donnée. Puis, la méthode B est étudiée en détails pour l'écriture des spécifications sous forme formelle et pour l'écriture des preuves automatiques de programmes. Les logiques de Hoare et de Dijkstra sont ensuite présentées pour la vérification formelle des programmes écrits sous formes de triplets. Dans le contexte des systèmes réactifs, le paradigme du Model-Checking est étudié pour la validation des propriétés de systèmes exprimées en logiques temporelles. En fin, une introduction aux méthodes de tests des logiciels est fournie.

Des exemples académiques et d'autres issus du monde industriel/réel sont donnés le long du cours pour illustrer les différentes notions abordées

Bibliographie :

- Jean-Raymond Abrial, The B Book - Assigning Programs to Meanings, Cambridge University Press, August 1996.
- C.A. Gunter and D.S. Scott, Semantic Domains, In Jan van Leeuwen, Editor, Handbook of theoretical computer science, Elsevier Science, Publisher, 1990 (pp. 633-676).
- Jacques Julliand, Vérifier, tester et concevoir des programmes en les modélisant, Vuibert, 2010 | 272 pages | 9782311000207.
- C. Baier and J-P. Katoen. Principles of model checking, MIT Press, 2008.
- E-M., Clarke, T-A., Henzinger, H. Veith, R. Bloem, Handbook of Model Checking, Springer, 1st ed. 2018 edition (May 18, 2018).
- A-P. Mathur, Foundations of Software Testing, Pearson Education, 2008.

Expected competencies:

Understand the development of reliable, secure and robust software, more particularly, in critical fields (such as transportation, Aeronautic and Energy). Be able to use formal methods to model the specifications, write proofs, check and prove the programs' correction and other proprieties useful in Software Engineering.

Prerequisites: Logic knowledge, programming, modelling and algorithmic.

Content:

This lecture presents the formal verification and validation methods both for traditional programs and reactive systems. After a brief introduction on the software quality expected, necessary mathematical tools are presented. Then, the B method is deeply studied for writing the specifications in a formal way and the automatic proofs of programs. Logics of Hoare and of Dijkstra are

also studied for formal verification and reasoning about the correctness of programs written in triple form. The paradigm of Model-Checking for the validation of properties expressed in temporal logics is presented as well. Last, an overview of the methods of software testing is introduced.

Small examples and examples stemming from industrial/real world are provided to illustrate the different notions introduced in this lecture.

Recommended reading:

- J-R. Abrial, The B Book - Assigning Programs to Meanings, Cambridge University Press, August 1996.
- C.A. Gunter and D.S. Scott, Semantic Domains, In Jan van Leeuwen, Editor, Handbook of theoretical computer science, Elsevier Science, Publisher, 1990 (pp. 633-676).
- J. Julliand, Vérifier, tester et concevoir des programmes en les modélisant, Vuibert, 2010, 272 pages, 9782311000207.
- C. Baier and J-P. Katoen. Principles of model checking, MIT Press, 2008.
- E-M., Clarke, T-A., Henzinger, H. Veith, R. Bloem, Handbook of Model Checking, Springer, 1st ed. 2018 edition (May 18, 2018).
- A-P. Mathur, Foundations of Software Testing, Pearson Education, 2008.