Logiciels Sûrs *Reliable Softwares*			
Code ECUE *Course code*: **LOS**		**UE :** UE5-2y	
Département *Department*	IA	**Cours** *Lectures*	10h00
Coordonnateurs *Lecturers*	A. Hadj-ali	**T.D.** *Tutorials*	10h00
Période *Year of study*	A3	**T.P.** *Laboratory sessions*	
Semestre *Semester*	S5	**Projet** *Project*	
Evaluation *Assessment method(s)*	1 écrit	**Non encadré** *Unsupervised*	
Langue d'instruction *Language of instruction*	Français	**Horaire global** *Total hours*	20h00
Type de cours *Type of course*	Obligatoire	**Travail personnel** *Homework*	07h00
Niveau *Level of course*	Second cycle universitaire *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