Université du Québec en Outaouais | **Département d'informatique et d'ingénierie**

Sigle : INF1763 Gr. 01
Titre : Techniques et outils professionnels de développement logiciel
Session : Automne 2026 Horaire et local
Professeure : Elouasbi, Samir

1. Description du cours paraissant à l'annuaire :

Objectifs
Au terme de ce cours, l'étudiant.e sera en mesure d'appliquer des méthodes de développement agile, à l'aide d'outils modernes pour le contrôle de version, la gestion des dépendances, ainsi que les tests et le déploiement automatisé.

Contenu
Méthodes de développement et de gestion de projet agiles. Travail d'équipe, revues de code. Réingénierie (refactoring). Contrôle de version (git, github), gestion des dépendances (Maven, Gradle, etc.). Environnements de développement et de production, outils de virtualisation et conteneurs. Construction et déploiement automatisés (make, ant, etc.), gestion de la configuration. Automatisation des tests et intégration continue (Jenkins, Circle CI, etc.). Approche DevOps. Ce cours comporte des séances obligatoires de travaux dirigés (TD).

Descriptif – Annuaire

2. Objectifs spécifiques du cours :

Méthodologie et Développement Agile

Au terme de cette section, l'étudiant.e sera capable de :

•Expliquer et appliquer les principes fondamentaux des méthodes agiles

•Mettre en œuvre un processus de développement agile en équipe

•Conduire des revues de code (Code Reviews)

•Appliquer des techniques de réingénierie (Refactoring)

Outils de Contrôle de Version avec Git et GitHub

Au terme de cette section, l'étudiant.e sera capable de :

•Maîtriser les commandes de base et avancées de Git

•Gérer des branches de développement de manière efficace

•Collaborer sur un projet en utilisant GitHub

Gestion des Dépendances

Au terme de cette section, l'étudiant.e sera capable de :

•Gérer les dépendances d'un projet avec Maven ou Gradle

Environnements de Développement et Déploiement

Au terme de cette section, l'étudiant.e sera capable de :

•Configurer et utiliser des environnements de développement virtuels

•Automatiser la construction (build) d'une application

•Mettre en place une chaîne d'intégration et de déploiement continus (CI/CD)

Tests Automatisés et Approche DevOps

Au terme de cette section, l'étudiant.e sera capable de :

•Écrire et exécuter différents types de tests automatisés

•Intégrer la culture et les pratiques DevOps

3. Stratégies pédagogiques :

Les formules pédagogiques suivantes seront utilisées :

Séances de cours en présentiel comprenant :
- Cours magistral avec démonstrations pratiques des outils
- Travaux dirigés (TD) obligatoires en laboratoire informatique
- Projet de développement en équipe utilisant les méthodologies agiles

Ce cours est à caractère pratique et appliqué.
Les méthodes d'apprentissage privilégiées pour la transmission des connaissances seront :
- **Apprentissage par la pratique** : manipulation directe des outils de développement
- **Pédagogie active** : travaux en équipe, revues de code collaboratives
- **Projet intégrateur** : développement d'une application complète du début à la fin
- **Séances de laboratoire** : configuration d'environnements, mise en place de pipelines CI/CD

Les attentes sont que les étudiant(e)s investissent au moins 90 heures de travail personnel en plus des 45 heures de cours et des heures de travail sur les projets d'équipe.

Équipement requis :
Les étudiant(e)s qui s'inscrivent à ce cours doivent s'assurer qu'ils ont :
- Un ordinateur portable avec au moins 8 GB de RAM
- Système d'exploitation : Windows 10/11, macOS, ou Linux
- Connexion Internet stable

4. Heures de disponibilité ou modalités pour rendez-vous :

Disponible sur rendez-vous.

5. Plan détaillé du cours sur 15 semaines :

Semaine	Thèmes	Dates
1	**Introduction au développement logiciel moderne et à l'approche DevOps** • Cycle de vie du développement logiciel et évolution des pratiques de développement. • Introduction à DevOps : collaboration, automatisation, intégration et rétroaction continue. • Présentation du cycle DevOps et des principaux outils du cours.	9 septembre 2026 **(Non présentiel)**

	Projet : Présentation du projet et des quatre mini-projets cumulatifs.	
2	**Méthodes agiles et gestion de projet** • Manifeste Agile, développement itératif et incrémental. • Scrum et Kanban : rôles, artefacts, sprints, flux de travail et amélioration continue. • Backlog, user stories, critères d'acceptation, estimation et suivi de l'avancement. **Projet :** Préparation du mini-projet 1 : constitution des équipes, définition des fonctionnalités et création du backlog initial.	16 septembre 2026
3	**Contrôle de version avec Git** • Principes du contrôle de version et fonctionnement d'un dépôt Git. • Working directory, staging area, commits, historique et HEAD. • Commandes fondamentales et bonnes pratiques de versionnement. **Projet :** Mise en place du dépôt Git du mini-projet 1.	23 septembre 2026
4	**GitHub et développement collaboratif** • Dépôts distants, branches, fusion et résolution de conflits. • GitHub : Issues, Pull Requests et protection des branches. • Workflow collaboratif et stratégies de branches. **Projet :** Application de GitHub et du workflow collaboratif au mini-projet 1. **TD 1 – Git : exercices sur les dépôts, commits, historique et restauration.**	30 septembre 2026
5	**Revues de code et réingénierie (refactoring)** • Qualité logicielle, dette technique, code smells, cohésion et couplage. • Principes et techniques courantes de refactoring. • Revues de code : critères d'analyse, commentaires, approbation et demandes de modification. **Projet :** Suivi du mini-projet 1 et présentation du mini-projet 2. **TD 2 – GitHub et collaboration : branches, fusions, conflits et Pull Requests.**	7 octobre 2026
6	**Semaine d'études**	14 octobre 2026
7	**Construction automatisée et gestion des dépendances** • Compilation, build, packaging et artefacts. • Présentation de Make, Ant, Maven et Gradle. • Maven : pom.xml, dépendances, plugins et cycle de vie. **Projet :** Remise du mini-projet 1. Mise en œuvre du mini-projet 2. **TD 3 – Revue de code et refactoring : analyse et amélioration d'un code existant.**	21 octobre 2026
8	**Tests logiciels et automatisation des tests** • Tests unitaires, d'intégration, système et d'acceptation. • JUnit : assertions, scénarios de test et organisation des tests.	28 octobre 2026

	• Couverture de code et intégration des tests au build Maven. **Projet :** Intégration progressive des tests automatisés au projet. **TD 4 – Maven et dépendances : configuration du projet, build et génération d'un artefact.**	
9	**Environnements et gestion de la configuration** • Environnements de développement, test, préproduction et production. • Fichiers de configuration, paramètres et variables d'environnement. • Gestion des secrets et séparation entre code et configuration. **Projet :** Remise du mini-projet 2. Présentation du mini-projet 3. **TD 5 – Tests automatisés : écriture et exécution de tests JUnit avec Maven.**	4 novembre 2026 **(Non présentiel)**
10	**Virtualisation et conteneurisation avec Docker** • Machines virtuelles et conteneurs : principes, différences et cas d'utilisation. • Docker : images, conteneurs, Dockerfile et registres. • Ports, volumes, réseaux et introduction à Docker Compose. **Projet :** Conteneurisation de l'application du mini-projet 3.	11 novembre 2026
11	**Intégration continue** • Principes et objectifs de l'intégration continue. • Pipelines, jobs, stages, runners/agents, logs et artefacts. • Automatisation du build et des tests; présentation de Jenkins, CircleCI et d'un outil de CI utilisé dans le cours. **Projet :** Présentation du mini-projet 4 et intégration de la CI au projet. **TD 6 – Docker : création d'une image et exécution d'une application conteneurisée.**	18 novembre 2026
12	**Livraison et déploiement continus (CI/CD)** • Distinction entre intégration continue, livraison continue et déploiement continu. • Automatisation du packaging, de la création d'image et du déploiement. • Gestion des artefacts, environnements, secrets et stratégies de retour arrière. **Projet :** Remise du mini-projet 3. Poursuite du mini-projet 4.	25 novembre 2026 **(Non présentiel)**
13	**Approche DevOps et intégration globale** • Culture DevOps : collaboration, automatisation, responsabilité partagée et amélioration continue. • Intégration des pratiques du cours dans une chaîne DevOps complète. • Introduction à l'observabilité, à l'Infrastructure as Code et à DevSecOps. **Projet :** Préparation de la validation finale du projet. **TD 7 – CI/CD : mise en place d'un pipeline automatisant le build et les tests.**	2 décembre 2026
14	**Validation et démonstration du projet intégrateur** • Démonstration du workflow Git/GitHub, du build, des tests et de la conteneurisation.	9 décembre 2026

	• Exécution et validation du pipeline CI/CD. • Questions techniques sur les choix réalisés et la contribution des membres de l'équipe. **Projet :** Remise du mini-projet 4 et du projet complet.	
15	**Examen final**	16 décembre 2026

6. Évaluation du cours :

L'évaluation vise à mesurer à la fois la compréhension des concepts et la capacité à appliquer les méthodes et outils du cours dans un contexte de développement logiciel collaboratif.

Évaluation	Description	Pondération
Mini-projet 1	Agile et développement collaboratif : backlog, user stories, Git, GitHub, branches, Issues et Pull Requests.	10 %
Mini-projet 2	Qualité et automatisation du build : revue de code, refactoring, Maven, dépendances et génération d'un artefact.	15 %
Mini-projet 3	Tests, configuration et conteneurisation : tests automatisés, environnements, variables de configuration et Docker.	15 %
Mini-projet 4	Pipeline DevOps complet : intégration des livrables précédents, CI/CD, documentation et démonstration technique finale.	20 %
Examen	Évaluation individuelle portant sur l'ensemble des notions et leur application.	40 %
Total		**100 %**

Projet intégrateur

Les quatre mini-projets sont cumulatifs et constituent les étapes d'un seul projet logiciel. Le mini-projet 4 correspond au livrable final et à l'intégration complète de la chaîne DevOps. La séance de validation du projet est obligatoire et fait partie de l'évaluation du mini-projet 4. Dans les travaux d'équipe, la note peut être ajustée individuellement lorsque la contribution observable d'un membre diffère significativement de celle de l'équipe. L'historique Git, les Pull Requests, les revues de code, la documentation et la démonstration peuvent être utilisés pour apprécier la contribution.

7. Politiques départementales et institutionnelles :

- Politiques relatives à la tenue des examens
- Note sur le plagiat et les fraudes
- Politique relative à la qualité de l'expression française écrite chez les étudiants et les étudiantes de premier cycle à l'UQO
- Absence aux examens : cadre de gestion, demande de reprise d'examen (formulaire)

Tolérance ZÉRO en matière de violence à caractère sexuel.
Le Bureau d'intervention et de prévention en matière de harcèlement (BIPH) a pour mission d'accueillir, soutenir et guider toute personne vivant une situation de harcèlement, de discrimination ou de violence à caractère sexuel. Le BIPH oriente ses actions afin de prévenir les violences à caractère sexuel pour que nous puissions étudier, travailler et s'épanouir dans un milieu sain et sécuritaire.
Vous vivez ou êtes une personne témoin d'une situation de violence à caractère sexuel ? Vous êtes une personne membre de la communauté étudiante ou une personne membre du personnel, autant à Gatineau qu'à Ripon et St-Jérôme, l'équipe du BIPH est là pour vous, sans jugement et en toute confidentialité.
Ensemble, participons à une culture de respect.
Pour de plus amples renseignements consultez UQO.ca/biph ou écrivez-nous au Biph@uqo.ca

8. Principales références :

Livres recommandés :
•HUMBLE, Jez, FARLEY, David. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional, 2010 (ISBN 978-0321601919).
•KIM, Gene, DEBOIS, Patrick, WILLIS, John, HUMBLE, Jez. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press, 2016 (ISBN 978-1942788003).
•CHACON, Scott, STRAUB, Ben. Pro Git. 2e édition, Apress, 2014 (ISBN 978-1484200773). Disponible gratuitement en ligne : https://git-scm.com/book
•MARTIN, Robert C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008 (ISBN 978-0132350884).
Ressources techniques et documentation :
•FOWLER, Martin, BECK, Kent. Refactoring: Improving the Design of Existing Code. 2e édition, Addison-Wesley Professional, 2018 (ISBN 978-0134757599).
•NEWMAN, Sam. Building Microservices: Designing Fine-Grained Systems. 2e édition, O'Reilly Media, 2021 (ISBN 978-1492034025).
•BURNS, Brendan, BEDA, Joe. Kubernetes: Up and Running. 3e édition, O'Reilly Media, 2022 (ISBN 978-1098110208).

9. Page Web du cours :

https://moodle.uqo.ca