LOG8430: Architecture logicielle et conception avancée

Présentation de Cours

Hiver 2017

Fabio Petrillo
Chargé de Cours

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Introduction

- Chargé de course et chargé de lab.
- Objectifs de l’informatique
- Objectifs du cours
- Programme de cours
- Évaluation
- Exercice de réchauffement

Chargé de cours

Fabio Petrillo

http://fabiopetrillo.com

fabio@petrillo.com

fabio.petrillo@polymtl.com

@petrillofabio

Local M-4221

Chargé de cours - Fabio Petrillo

- Diplôme d'ingénieur en génie électrique
- Maîtrise en génie logiciel
  - Méthode agile pour l'industrie du jeux vidéo
- **Doctorat en génie logiciel**
  - Analyse, compréhension et visualisations des programmes
  - Débogage - Swarm Debugging
- Plus de 20 ans d'expérience en génie logiciel
  - **Architecte logiciel au Brésil (Government)**
  - Gestionnaire
  - Expérience en systèmes complexes
- Chargé de Course et Lab à Poly (**LOG8430**, INF3710, et **LOG8371**)
- Research Associate (Postdoc) at Concordia University (Prof. Wahab)
  - Logging and Monitoring improvements at Ubisoft (MITACS)

Chargé de labo

Manel Abdellatif

manel-2.abdellatif@polymtl.ca

Étudiante de Doctorat à Poly

Expert en migration vers service

Objectifs

- Présenter les principaux types d’architectures logicielles et de critères de choix d’un type d’architecture
- Présenter des approches basées sur des patrons de conception pour la réalisation d’applications (distribuées)
- Donner un aperçu des approches émergentes de conception logicielle

À la fin du cours, l'étudiant se en mesure de

- **Concevoir** l'architecture d'un logiciel en **choisissant** et en **justifiant** les architectures, les patrons de conception et les techniques de conception avancée utilisés (entre autres à la méta-programmation, la programmation par composants, les patrons d'architecture distribuée, les architectures orientées services) ;
- Concevoir les composantes de l'architecture d'un logiciel (classes orientées-objets, composants logiciels ou services logiciels) à l'aide de techniques avancées et à partir de patrons d'architecture distribuée parmi les plus utilisés et de justifier ses choix ;
- Juger de la conception de l'architecture d'un logiciel et de suggérer des changements à cette architecture.

Situation du cours

LOG8430 termine une **chaîne** de cours :

- INF1005C - programmation **procédurale**
- INF1010 - programmation **orientée objets**
- LOG2410 - **conception** logicielle

Situation du cours

- LOG8430 est aussi un cours **gradué**
- Être à l’aise pour lire et écrire (en l’anglais)
- Faire de la recherche
  - Pensée critique
  - Créer des connaissances
- Concepts avancés
  - Être à l’aise avec les concepts OO
  - Être à l’aise avec des langage de programmation
  - Être à l’aise avec installation des outils (se **débrouiller**)
  - **Être à l’aise pour prendre des décisions et les soutenir**

Programme de cours

Cours	Contenus
00	Plan de cours
01 - 04	Fondements de architecture logicielle Modélisation, concepts de la POO Patrons des projets et les styles architecturaux Analyse et évaluation de la qualité logicielle Conception d'architecture logicielle
05 - 08	Méta-programmation Cadriciels (Frameworks) et platforms Systèmes distribués; Architecture orientée services (SOA), Microservices, Serverless
09 - 10	Reactive programming, distributed streaming, message-driven applications, data intensive applications, Cloud Computing Architecture...
11 - 12	Présentations des Papier, Présentation TP3 et Révision

Principaux manuel du cours essentielles

- *Software systems architecture: working with stakeholders using viewpoints and perspectives* , 2nd ed.; Rozanski, Nick. Addison-Wesley, 2011.
- Just Enough Software Architecture: A Risk-Driven Approach by George H. Fairbanks, Marshall & Brainerd, 2010.
- *Designing Software Architectures: A Practical Approach*, Humberto Cervantes; Rick Kazman, Addison-Wesley Professional, 2016.
- *Software and Systems Architecture in Action*, Raghvinder S. Sangwan; Auerbach Publications, 2014.
- *Documenting software architectures: views and beyond*, Paul Clements, 2nd; Addison-Wesley, 2011.
- *Patterns of Enterprise Application Architecture*, Martin Fowler; Addison Wesley Professional, 2002.

Outils

- Moodle
- Slack - https://log8430.slack.com
- Trello - https://trello.com
- Overleaf - https://www.overleaf.com
- Mendeley - https://www.mendeley.com/
- Git - https://github.com/
- CASE/Modelling Tools - Modelio, Papirus, ArgoUML, Visual Paradigm, Enterprise Architect, Archi Tool, ….
- Static analysis tools: Better Code Hub, Ptidej, Understand, SonarQube, CodeScene, ….

Important

- Toutes les matières présentées pendant les séances de cours ou de lab. sont contrôlable
- **Prenez vos notes de cours; les diapos ne seront pas disponibles.**
- Les travaux remis en retard seront pénalisés de 10% par jour de retard
- Toute demande d’examen différé doit passer par le bureau des affaires académiques

Important

Les règlements relatifs au plagiat s’appliquent en tout temps

Évaluation du cours

- Travaux Pratiques 30%
- Papier de Recherche 30%
- Examen Finale 30%
- Quiz et participation 10%

Travaux Pratiques

- En équipe (3 - 4 max)
- Objectifs
  - Compréhension et analysis
  - Conception
  - Implementation
- 3 rapports (TP1 - 3) et une présentation finale (TP4)
- **Pondération 30%**

Organisation des travaux pratiques

TP	Sujet	Pondération	Remise
TP1	Étude et analyse d'architecture	30%	13/02/2018
TP2	Conception and implementation	30%	20/03/2018
TP3	Migration - Arch. orienté à services	30%	15/04/2018
	Présentation finale (en classe)	10%	16/04/2018

Papier de recherche

- Papier de recherche sur un sujet de architecture logicielle
- **Révision systématique de la littérature (SLR)**
- **Individuelle ou en groupe**
- Nombre des pages: **10 + [(nombre d'auteurs - 1) x 5] pages**
  - Individuelle - 10 pages (full conference paper)
  - 2 auteurs - 10 pages (full conference paper)
  - 3 auteurs - 15 pager (journal paper)
  - 4 auteurs - 20 pager (top journal paper)
- Format finale **Latex/Doc IEEEtrans**
  - **https://www.ieee.org/conferences_events/conferences/publishing/templates.html**
- Langue: **anglais** (préférable) ou français
- Outils d'édition: Overleaf (http://www.overleaf.com)
- **Soumission à une vrai conférence** (pas obligatoire)
- **Pondération 30%**

Structure de base pour le papier

- Abstract: une synthèse de votre papier pour guider le lectures/attirer pour lire le papier
- Introduction: problème ciblé, motivation, proposition, organization du papier
  - avec une (ou des) question(s) de recherche
- Background: les concepts/contexte nécessaire pour comprendre votre papier (references)
- Travaux connexes
- Approche/méthodologie: votre approche pour le problème ciblé
- Résultats: les résultats par rapport l'application de votre approche
- Discussion
- Threats to the validity
- Conclusion: les conclusions et travaux futures

Papier - Points évalués

- Importance de la **contribution** du papier et avantages que d'autres peuvent tirer de la contribution: pourquoi la contribution et les avantages sont-ils importants?
- **Originalité** du travail: quelles nouvelles idées, discussions ou approches sont introduites?
- **Validité** du travail présenté: comment les chercheurs et les praticiens peuvent-ils utiliser les résultats avec confiance?
- **Clarté** de présentation;
- **Travaux antérieurs** pertinents: le travail préalable a-t-il fait l'objet d'un examen adéquat?

Choose one of the research topics (Main ICSA Themes)

- Microservices, Reactive programming, High scalable architectures
- Up-front architecture and agile development
- Architecting IoT systems
- Architecting Machine/Deep learning systems
- Architecting Blockchain systems
- Component based software engineering and architecture design
- Architecture evaluation and quality aspects of software architectures
- Automatic extraction and generation of software architecture descriptions
- Re-factoring and evolving architecture design decisions and solutions
- Architecture frameworks and architecture description languages
- Software architecture knowledge management
- Software architecture for legacy systems and systems integration
- Software architects roles and responsibilities

Remises pour le papier de recherche

Remise	Livrable	Pondération	Date de Remise
R1	Définition du sujet	5%	19/01/2018
R2	Introduction, Background, Travaux connexes et méthodologie	15%	02/02/2018
R3	Results, Discussion, Threats to Validity, Abstract et Conclusion (brouillon complet) - **Optionnelle**	0%	16/02/2018
RF-A RF-B RF-C	**Version finale (A,B,C)**	**A - 70%** B - 60% C - 50%	**16/03/2018** 06/04/2018 20/04/2018
	Présentation Finale	**10%**	**26/03 et 09/03/2018**

Examen Final

- Individuel
- **Pondération 30%**
- **Sans consultation**
- 4~5 questions obligatoire et des questions optionnelles
- Questions similaire seront discutés en classe
- Élaboration des questions par les étudiantEs
  - a chaque cours peut demander aux présentes à proposer de questions avec la réponse
  - sujet de la séance ou les prochaines (étudient en avance!!!)
  - je vais utiliser les bonnes questions dans l'examen
  - **l'auteur recevra les points sans faire la question!**

Quizzes et participation en classe

- Individuel et collaboratif
- Un quiz de révision et des exercices
- Collecter la participation en classe
- Feedback en classe
- **Pondération 10%**
- Questions de révisions et préparation pour l'examen
- https://socrative.com

Quiz 00 - Simple Survey

- https://socrative.com
  - Login Student
- Room: LOG8430
- Answer the questions

Exercice en classe (évaluation de connaissance)

Modéliser un système de **gestion de bibliothèque** à gérer un réseau de centaines de bibliothèques sur quatre continents avec des centaines de milliers d'utilisateurs simultanés ayant la possibilité d'emprunter dans n'importe quelle bibliothèque de votre réseau (proposez votre processus pour emprunter des livres). Décrivez vos choix : **style architectural, cadriciels, le mécanisme de persistance, etc. Utilisez les patrons de conception adéquats (ex. Repository).** Important: (1) utilisez les diagrammes **UML** pour modéliser votre architecture en utilisant tous les diagrammes nécessaires pour décrire votre système (ex: les diagrammes de cas d'utilisation, de classe, de composant, de déploiement, d'état, de séquence, etc). (2) Votre solution doit privilégier **la scalabilité horizontale**.