UNIVERSITÉ BISHOP'S UNIVERSITY

Computer Science Department

CS 410/CS 560 - Software Engineering

Course outline

Winter 2020

Professor:

Dr. Mohammed Ayoub Alaoui Mhamdi
Email: Mohammed.Ayoub.Alaoui.Mhamdi@ubishops.ca
Office: Bishop's University, HAM 161
Office hours: To be determined

Schedule:

Lecture:

MW	4:00 pm to 5:29 pm	HAM 301
WF	8:30 pm to 9:59 pm	BWH

Description

Objective: To learn the concepts of software architecture and the object-oriented software design with the help of UML.

Content: In this course, students learn the principals involved in the analysis of large software systems, Topics also include: design patterns, anti-patterns and General Responsibility Assignment Software Patterns (GRASP). Participants also gain hands-on experience using a case tool to draw most of the Unified Modelling Language (UML) diagrams necessary to support object oriented analysis and design activities. Students learn how to verify and validate models by checking consistency of UML models also will be introduced to semantics to understand form semantics for UML. The students are supposed to work on a big project in order to apply the concepts of software architecture seen in class.

Credits: 3

Organization: Three hours of lecture per week
Six hours of personal work per week

1 Overview

1.1 Context

As software systems grow in size and complexity, their design problem extends beyond algorithms and data structures to issues of system design. These issues the software architecture level of software design are becoming increasingly important to the practicing software engineer. The topics covered in this course include, software design process; object-oriented design using UML; design patterns; programming environments.

1.2 Learning objectives

At the end of this educational activity, the student will be able to:

1. Understand and apply software architecture and design techniques;
2. Select and use appropriate design patterns;
3. Utilize and evaluate computer aided software engineering tools;
4. Model the analysis and design of software component using UML;
5. Verify and validate software engineering models.

1.3 Technical competences

1. Express the analysis and design of an application using UML;
2. Specify functional semantics of an application using OCL;
3. Specify and evaluate software architectures;
4. Select and use appropriate architectural styles;
5. Understand and apply object-oriented design techniques;
6. Select and use appropriate software design patterns;
7. Understand and perform a design review.

1.4 Transversal competences

1. Acquire abilities to read and review research papers and articles and use research methods to solve related domain problems;
2. Acquire abilities to prepare structure, write documents, and report with the right level of formality and technical detail;
3. Acquire abilities to communicate and transmit knowledge in an oral manner;
4. Acquire abilities to work effectively in a team to successfully design and implement a software system.

1.5 Theoretical and practical contents

a. Theoretical contents

- SW Design Concepts
- UML and Analysis
- SW Architecture
- SW Design

b. Practical contents

A group of assignments and a course problem will make up the practical component of the course are shown in Table below.

Practical Component	**Objectives**
Assignment 1 Reading activity	1. Understand the concept of software engineering; 2. Understand the different models of software lifecycle.
Assignment 2	The learning involves the development of initial design documents to ensure that the team has a solid grasp of the problem domain in general, along with the client's requirements that are more specific.

	Deliverables: - Object Oriented Analysis - UML Class Model Diagram
Assignment 3	In this phase, the student will begin to implement some of basic functionality for the system. Deliverables: - Implementation source code - UML Class Model Diagram Update
Assignment 4: UML State Chart Diagram and OCL	• Produce state chart diagram modelling for a home security systems • Loyalty Program Information System.
Assignment 5	In this phase, the students will complete the learning process by designing advanced patterns.

1.6 Learning plan

Teaching activity	**Methodology**	**Face-to-face hours**	**Self study hours**
Problem based learning method	Methods of instruction include lecture, class discussion, group and/or individual assignments including project assignments and specific assignments related to specific important topics in courses.	48 hours	44 hours

	Course may be taught as face-to-face, or hybrid.		
Total	92 hours	48 hours	44 hours

1.7 Detailed content

Weekly detail

*****(T) for theoretical, (InP) for in class practise, (P) for out class practise.

WEEK	Type (T, P, InP)	UNIT	Required Readings	Self study	Hours
Week 1 and week 2	T	1. Introduction; 2. Software Engineering Management; 3. The Software Life Cycle Revisited. 4. Configuration Management	Software Engineering: Principles and Practice. Hans van Vliet	6	3
	P	**Assignment 1**			
Week 3	T	5. Object Oriented Analysis	UML Specification, Chapter 1	4	3
	T	6. Review of UML			
	P	7. UML Class Models			
Week 4	P	7. Design Studies	[5]	6	2.30
	P	8. Library Example (UML)			

	P	9. Formal Specification Exercise			
	P	**Assignment 2**			
Week 5	T	10. Object Constraint Language OCL	[5]	3	2.30
	P	11. Library Example (OCL)			
Week 6	T	12. Behaviour Modelling	On Visual Formalisms, David Harel	4	2
	P	13. Modelling with State charts			
	P	**Assignment 3**			
Week 7 and week 8	T	14. SW Architecture	An Introduction to Software Architecture, David Garlan	7	5
	P	15. SW Architecture Example			
	T	16. Architectural Views	The 4+1 View Model of Architecture, PHlLlPPE B. KRUCHTE, Rational Software Other Views Features, non-functional requirements, bug reporting, context, utility		
Week 9 and week 10	T	17. Architectural Styles and Non-Functional Requirements	Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach	7	4

	T	18. Taxonomy of Connectors	Towards a Taxonomy of Software Connectors, Mehta		
Week 11	T	19. Architecture Description, Language ADLs, (ACME Project)	A Classification and Comparison Framework for Software Architecture Description Languages, Medvidovic	6	4
	T	20. Refinement (Complexity and Abstraction)			
	P	**Assignment 4**			
Week 12	T	21. Middleware Architecture of Distributed Systems (Multiple Computers, Heterogeneous) Example: Client-Server, Database Server, Client Web Browser, Business logic	Software Engineering and Middleware: A Roadmap, Emmerich	6	2
Week 13	T	22. Components (Bottom-Up Design)	- Component Software, Szyperski - Software Component Models, Lau and Wang	6	2.30
	P	23. Actual Process of Designing Objects	Heuristics and Coffee, Chapter 11		
	T and P	24. Object Design	From OOA to OOD - Intermodal Consistency - From Analysis to Design - System Design - Abstraction mechanisms - Collaboration-based design		
Week 14	T	Design Patterns	Describing Design Patterns, Gamma Paper	4	2.30

	T	Design Principles	Granularity, Martin Engineering Notes		
	P	**Assignment 5**			
	P	Design Reviews			

2 Organization

2.1 Teaching method

A week includes two hours and half of lectures. Most classroom presentations will be done using slides available on Moodle. Throughout the session, the student will implement the concepts seen in class through five assignments, a final project, and a final exam.

2.2 Evaluation

Assignments: 40%

Final project: 30%

Final exam: 30%

Unless otherwise stated, the assignments can be done in teams of four students. Special instructions will be given for each assignment. The submission of assignment must be performed using Moodle. Subjects will be available on Moodle. The delivery of work is done before midnight of the due day. Failure to submit an assignment before the deadline will result in a penalty of 10% per day late, including Saturday and Sunday.

No assignment can be submitted by email. Plagiarized or undelivered assignment will be automatically attributed the grade zero.

For the final project, a team of four students can do it. The subjects of final project will be available on Moodle by the second week of February.

3 Course Material

The course slides will be available on Moodle.

No manual is required.

4 Bibliography

[1] Hans van Vliet. Software Engineering: Principles and Practice. Wiley; 3rd edition, 2008.

[2] Carliss Y. Baldwin, Kim B. Clark. The MIT Press; 4th Printing edition, 2000.

[3] Richard N. Taylor, Nenad Medvidovic, Eric Dashofy. Software Architecture: Foundations, Theory, and Practice. John Wiley & Sons, 2009.

[4] Kai Qian, Xiang Fu, Lixin Tao, Chong-wei Xu. Software Architecture And Design Illuminated. Jones & Bartlett Learning; 1st edition 2009.

[5] Timothy Lethbridge, Robert Laganiere. Object-Oriented Software Engineering: Practical Software Development Using UML and Java: Practical Software Development using UML and Java, Second Edition. McGraw-Hill Education / Europe, Middle East & Africa; 2th edition, 2004.