Department of Computer Science and Software Engineering
Concordia University

SOEN 6461 Software Design Methodologies
Course Outline Fall 2015

Contact Information

Section SS Course Instructor: Dr Greg Butler (gregb@encs) EV 3.219
Course Web Site: http://users.encs.concordia.ca/~gregb/home/soen6461-f2015.html

(Official) Course Description *(from the Graduate Calendar)*

SOEN 6461 Software Design Methodologies (4 credits) Prerequisite: COMP 5541.

Introduction to software design processes and their models. Representations of design & architecture. Software architectures and design plans. Design methods, object-oriented application frameworks, design patterns, design quality and assurance, coupling and cohesion measurements, design verification and documentation. A design project. **Note**: Students who have received credit for COMP 6471 before September 2011 may not take this course for credit.

Course Objectives

This course is a 4-credit course with 3 contact hours (2.5 class hours) of lecture, and 2 hours of lab. Through the lectures and the design project, you will learn the different perspectives of software design in the context of the Software Engineering discipline: basic principles, formalisms, methodologies, documentation, and design evaluation.

You should expect to **average** a total of 10–12 hours per week on this course. Some individual weeks it may be much higher depending on deadlines fo assignments, quizzes, and examinations. So plan your time accordingly.

Prerequisite Knowledge

Object-oriented programming in either Java or C++; Knowledge of object-oriented concepts such as class, object, inheritance, polymorphism, identity, state, behaviour, delegation, abstract class, generic class; Be able to develop, test, and debug to construct a working object-oriented program of approximately 1000 lines of code; Be able to use an IDE such as Eclipse to develop programs; Be able to perform unit testing using a framework such as JUnit.

Be able to use standard object-oriented libraries for data structures and user interfaces.

Introductory knowledge of the software process (feasibility, requirements, design, implementation, test, deployment, maintenance, and evolution) and related documents; Working knowledge of UML notation and tools; Knowledge of UML concepts such as element, class, object, attribute, operation, association, generalization, aggregation, composition, dependency, message, state, compound state, concurrent regions, transition, event, action, activity, swimlane, package.

Graduate Attributes

As part of both the Computer Science and Software Engineering program curriculum, the content of this course includes material and exercises related to the teaching and evaluation of graduate attributes. Graduate attributes are skills that have been identified by the Canadian Engineering Accreditation Board (CEAB) and the Canadian Information Processing Society (CIPS) as being central to the formation of engineers, computer scientists and information technology professionals. This particular course aims at teaching and evaluating three graduate attributes. The following is a description of these attributes, and how these attributes are incorporated in the course.

(1) Problem analysis is the ability to use appropriate knowledge and skills to identify, analyze, and solve complex engineering problems in order to reach substantiated conclusions. The design assignments in this course are defined in such a way that requires the students to analyze the problem at hand and determine for themselves exactly what needs to be done, and then determine how the design can be achieved.

(2) Design is the ability to design solutions for complex, open-ended engineering problems and to design systems, components or processes that meet specified needs with appropriate attention to health and safety risks, applicable standards, and economic, environmental, cultural and societal considerations. The design assignments in this course are presented in an open-ended fashion. The individual assignments provide a platform for designing at a smaller level.

(3) Communication skills are the ability to communicate complex engineering concepts within the profession and with society at large. Such abilities include reading, writing, speaking and listening, and the ability to comprehend and write effective reports and design documentation, and to give and effectively respond to clear instructions. In this course, each of the design assignments requires you to decide on the important information to communicate, and to create UML diagrams and text that clearly communicate the concepts in your design.

Resources

Textbook: We rely heavily on the textbook, so get a copy and read it thoroughly.

Craig Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, rd edition, Prentice-Hall, 2005.

Recommended Books: These are valuable references for any practicing software developer, so it is worthwhile to buy them. You will need to understand the Iterator and Strategy design patterns very well for your assignments, and to understand architecture for the third assignment.

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, *Pattern-Oriented Software Architecture, A System of Patterns*, Wiley, 1996.

Evaluation

Students are required to complete three design assignments (30%) two quizzes (20%), and a final exam (50%).

The design assignments are to be done **individually**.

Each quiz will be a closed book examination of at least one hour in duration. Each quiz will require you to answer questions about the lecture material, and about the designs in the design assignments that have already been completed..

The final examination will be a closed book examination of three hours in duration. The final examination will require you to create a small design as part of the examination, and to answer questions about the three design assignments and the lecture material.

You must pass the final examination in order to pass the course as a whole. You must pass the assignments in order to pass the course as a whole.

Your grade will depend on your performance in the assignments, quizzes and the final examination A poor performance in any single component may bring down your grade. There is no simple direct correlation between your total mark and the grade.

Code of Conduct and Academic Integrity: All students should become familiar with the University's Code of Conduct (Academic). The Expectations of Originality form has been created to ensure that all students in the Faculty of Engineering and Computer Science comply with principles of academic integrity prior to submitting coursework to their instructors for evaluation: namely reports, assignments, lab reports and/or software. This form must be attached to the front of all coursework submitted to instructors in the Faculty of Engineering and Computer Science. See https://www.encs.concordia.ca/current-students/forms-and-procedures/expectation-of-originality/

Design Assignments

There are three assignments were you must create, evaluate, and document a design. The designs are due approximately in Weeks 4, 8, and 12.

The deliverable is a design document of about five (5) pages as a pdf file, submitted to the ENCS electronic submission system. Your design document should have subsections:

1. a brief description of the problem (up to one half (0.5) page);
2. a concise description of the design (at least two (2) pages and at most four (4) pages), where you focus on presenting the most important parts of the design (whatever they may be);
3. a brief description of the major design decisions (one half (0.5) to one (1) page in length);
4. a brief description on how you reviewed the design (one half (0.5) to one (1) page in length), and which qualities of the design were considered during the review;
5. a glossary of important things (one half (0.5) to one (1) page in length), such as classes, objects, methods, algorithms, and data structures.

Write your design document for a reader audience at the level of a CS or SOEN graduate like yourself.

Communicating your design clearly is a very important skill that you must learn. Use UML diagrams where and when they are appropriate for communicating information about the design, but do not use them if they are not needed.

Marking Scheme

For each design document, the marking scheme will assign a mark out of 2 for each of the subsections listed above, for a total mark out of 10:

- 0 marks for a missing subsection, or very unclear section, or where the section content is not appropriate for the section (ie mis-placed);
- 1 mark for an informative subsection that does not have too many errors;
- 2 marks for a clear, informative, concise, almost error-free subsection.

Length Penalty: There is a penalty of 10% of the assignment mark for each page of your submission over 7 pages.

Late Penalty: There is a penalty of 10% of the assignment mark for each day that your assignment is late.

Lecture Schedule *(Tentative — subject to change)*

Week 1 — 14 September 2015: Course Outline; Life-long Learning for Software Development; Major Issues in Software Engineering; Intro to Design. Reading: Larman, Ch 1–7.

Week 2 — 21 September 2015: Intro to Design; Detailed Design: Collections, Maps, Algorithms, and Libraries; Domain Modeling. Reading: Domain Modeling Larman Ch 8–12. Design Evaluation and Analysis Ch 33.

Week 3 — 28 September 2015: Evaluating Designs. Reading: Larmen, Ch 33.

Week 4 — 5 October 2015: Object-Oriented Design; Functional Design; Modularity, Interfaces, Encapsualtion. Reading: Larman Ch 12, 14, 17.

Week — 12 October 2015: Thanksgiving holiday — No lecture.

Week 5 — 19 October 2015: Responsibility-Driven Design (responsibilities, GRASP, use case realization, visibility). Reading: Larman Ch 12, 14, 17, 18, 19.

Week 6 — 26 October 2015: Quiz 1.

Week 7 — 2 November 2015: Design patterns. Reading: Larman Ch 26, 36. Gamma Ch 1–2. Gamma patterns: Iterator, Strategy; Composite, Facade, Proxy; Factory Method, Template Method, Visitor; Observer, Command.

Week 8 — 9 November 2015: Software Architecture (Layers, MVC, Blackboard). Reading: Larman Ch 13. Buschmann Ch 1–2.

Week 9 — 16 November 2015: Documenting Architectures. Reading: Larman Ch 33–39.

Week 10 — 23 November 2015: Quiz 2.

Week 11 — 30 November 2015: Software Architecture Styles and Patterns. Reading: Buschman Ch 1–2.

Week 12 — 7 December 2015: Software Architecture Evaluation and Analysis. Reading: Larman Ch 33. R. Kazman, M. Klien, P. Clements, *ATAM: Method for Architecture Evaluation*, CMU/SEI-2000-TR-004, 2000.

Week 13 — Tuesday 8 December 2015: Software Frameworks.

Week 14+ — Final Examination

Format: multiple choice, one word answers, short answer questions; & a design question.

Material: Week 1 to Week 12 chapters of Larman. Ch 1–19, 25, 26, 33–39. Architectures: Layers, Blackboard; Design patterns: Iterator, Strategy, Composite, Facade, Proxy. Designs in Assignments 1–3 including UML and evaluation/review of design.

Advice: Connect the lectures to the design assignments; Think about your use of architectures, design patterns, and GRASP patterns in your assignments.