Concordia University
Department of Computer Science
and Software Engineering

Advanced Programming Practices
SOEN 6441 --- Fall 2011 --- Section U

Contact Information

instructor:	Joey Paquet, EV-3-221 paquet@cse.concordia.ca
web page:	newton.cs.concordia.ca/~paquet/wiki/index.php/SOEN6441U_fall_2011

Calendar Description

Problems of writing and managing code. Managing complexity: programming process. Pragmatic Programming. Coding conventions, software documentation. Software configuration management. Advanced debugging techniques: program tracing, dynamic inspection and tools. Testing: coding techniques for testing software. Multithreading concurrency and distributed programming. Multi-language programming. A project. Laboratory: two hours per week. Note: Students who have received credit for COMP 6441 may not take this course for credit.

Rationale

Most students coming out of introductory programming courses (e.g. COMP248) know how to write simple programs written in a specific programming language. Other more advanced courses (e.g. COMP249) show them how to use the more advanced features of the language, thus reaching for a complete understanding of the syntactical constructs of a specific programming language. Industrial programming requires a lot more diversified skills than simple mastery of language syntax. Industrial programmers have to know how manage the complexity of their coding activities, install and use libraries, create reusable, documented, fault free and fault tolerant code, and produce applications that are often written using different programming languages. This course aims at broadening the knowledge of the students to these concepts, techniques and tools that are complementary to what is taught in standard programming courses.

Objective

Improving the practical programming skills of students by emphasizing real-life aspects of programming that are not dealt with in regular introductory and advanced programming courses. Practical mastery of techniques and tools for the writing of superior quality code and complementary programming artifacts such as inline documentation, design patterns, and automated testing infrastructure.

Prerequisite knowledge

Although this course does not officially have course prerequisites, it is taken for granted that the students are already proficient programmers that master the object-oriented programming paradigm. The lectures, exercises, examinations, and completion of the project will necessitate pre-existing programming skills. This course is not meant to teach you how to program, but rather how to extend your programming skills.

Project

The project is to be tackled by teams of exactly 4 members. It is divided into 3 practical assignments related to the project. Each assignment includes the delivery of an operational subset (i.e. and increment or build) of the final project. The project consists of a large program whose development involves most of the topics discussed in the lectures. Each build is graded independently of the other builds.

Grading

Examinations (midterm: 20% + final: 30%)	50%
Project build 1	15%
Project build 2	15%
Project build 3	20%

Textbook

Absolute Java. Walter Savitch. Addison Wesley. Fourth Edition, 2009.