CMPT 373 - D100 Software Development Methods

Instructor(s): Brian Fraser

Calendar Objective/Description:
Survey of modern software development methodology. Several software development process models will be examined, as will the general principles behind such models. Provides experience with different programming paradigms and their advantages and disadvantages during software development.

Instructor's Objectives:
This course exposes students to modern software development practices. Several software best practices will be introduced. Students will gain experience with different programming methodologies and their advantages and disadvantages during software development.

The course is principally a laboratory course, with lectures, discussions, and project homework to supplement the laboratory work. Students will work in groups of roughly eight individuals on common projects, with groups assigned by the instructor. Each project will come with some specification of implementation language (generally Java or C++), development platform (such as Linux), and other technologies (such as BOOST, MySQL, or OpenGL). Students should be prepared to promptly learn new aspects of the project so they can contribute quickly (such as web or UI programming).

The weekly laboratory times are for mandatory project group meetings, including meetings and code reviews with the instructor. The primary goal of the laboratory work is to correctly follow and understand the development practices for the project; students are marked individually depending on their adherence to the model and contribution to the project. Projects will consist of multiple iterations and students will make weekly contributions.

Students should expect to participate in class discussions and to give an informal presentation regarding their project or assigned specialty.

Prerequisites:
CMPT 213 and (CMPT 276 or 275).

Topics:
- Best practices: design patterns, refactoring, language-specific issues
- Agile software development: such as Scrum, extreme programming, test-driven development
- Managing complexity and designing maintainable software
- Software-engineering tools and environments
- Software development process models: component-based development, iterative processes
- Requirements gathering and teamwork

Grading:
Project: 50% Reading Responses: 20% Exercises and in-class activities/quizzes: 30% To be confirmed in the first week of classes.

Required Books:

**Reference Books:**

Clean Code, Robert C Martin, Prentice Hall, 2009, 9780132350884

**Academic Honesty Statement:**

Academic honesty plays a key role in our efforts to maintain a high standard of academic excellence and integrity. Students are advised that ALL acts of intellectual dishonesty will be handled in accordance with the SFU Academic Honesty and Student Conduct Policies (http://www.sfu.ca/policies/gazette/student.html).