Department of Computer Science and Software Engineering
Concordia University

COMP 354 — Introduction to Software Engineering
Course Outline
Winter 2016

Web page: `http://users.encs.concordia.ca/~gregb/home/comp354-w2016.html`

1 Course Calendar Description

Software development process models (e.g. linear vs. iterative). Project management: roles, activities and deliverables for each software life cycle phase. Requirements management: analysis, elicitation, and scope. Architecture, design and the mapping of requirements to design and design to implementation. Traceability. Software quality assurance: verification, validation and the role of testing. Maintenance and evolution. Project. Lectures: three hours per week. Tutorial: one hour per week. Laboratory: two hours per week.

Prerequisite: COMP 352; ENCS 282

Course website: Check frequently the website for announcements, course material, assignments, etc.

IMPORTANT NOTE: In the event of extraordinary circumstances beyond the University's control, the content and/or evaluation scheme in this course is subject to change.

2 Course Objectives

This course is a 4-credit course with (2.5 class hours of lecture, 1 hour of tutorial and 2 hours of lab. Through these three types of instruction you are taught the different perspectives of Software Engineering discipline : basic principles, formalisms, tools and team work. You will be learning a disciplined process of developing software and practicing it in a small project.

You should expect to **average** a total of 10–12 hours per week on this course. For individual weeks it will be much higher depending on your role in the project and the phase of the project. So plan your time accordingly.

Recommended Books:

Roger Pressman, *Software Engineering: A Practitioner's Approach*, McGraw-Hill.

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

Web Page There will be a course web page. You will find additional information there.

3 Course Organization

Part 1: Lectures

The lectures are a key component of the course, and you are advised to attend the lectures regularly and attentively. Throughout the term you need to periodically refer to the recommended textbooks. The study of the textbooks is not a substitute for lecture attendance as the latter will provide complementary examples, alternative perspectives and insights. The course slides will be posted on the course website. It is recommended to review the slides before you come to class. Note that slides are not "lecture notes"; you are responsible for creating your own notes.

Part 2: Tutorials

Tutorials will take place every week starting the second week of semester. Tutorial attendance is strongly encouraged. The tutorials will reinforce the material seen during the lectures with examples and practical exercises.

Part 3: Lab

Labs will take place every week starting the second week of semester. The labs will provide instructions on tools and conduct various reviews related to the term projects. Additionally students can use the lab time to synchronize their work and to define the project plan for the next week. All students are expected to attend the labs at the specified time and place.

Part 4: Project

This is an important activity in the course — doing a team project. The most important aspect of the project is experiencing the process of software development in order to appreciate how the various phases interact, and to see the roles of the deliverables. The second most important aspect is to experience the group dynamics and interpersonal relationships of team work.

The project is to be undertaken in teams of about 6 (randomly assigned) members and consists of building a challengingly large Java application. Each team has three roles: coders, testers and documenters. Each role is fulfilled by 2-3 students. The project is divided into three iterations or increments. After each iteration, the resulting application will be demonstrated in the lab. Team members are required to rotate roles after each iteration.

In addition, each student is required to keep a diary of his/her individual project activities and to submit their diary at each stage of the project.

The average work load per team member is expected to be 10 hours per week. The detailed project description will be posted on the course web page.

4 Evaluation Scheme

The course grade is based on a clear pass (50%) in each of (i)quizzes and (ii) project work. There will be two quizzes each worth 22.5% of the total mark, and project work worth 45% (essentially 15% per phase) for team work, plus 10% for individual effort in the project. The project work is evaluated in all three phases — based on the deliverables (meeting deadlines and quality levels appropriate to a student project), reviews (class presentations and discussions), and the final demonstration of the product (completeness with respect to the requirements, error free code, etc;). The entire team is awarded points in the project.

Your grade will depend on both your performance in the quizzes and your performance in the project. 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.

5 Academic Honesty

Violation of the Academic Code of Conduct in any form will be severely dealt with. This includes copying (even with modifications) of program segments. You must demonstrate independent thought through your submitted work. The Academic Code of Conduct is available at: http://www.concordia.ca/students/academic-integrity/code.html

6 Important Dates to Remember

DNE Date: 19 January 2016

DISC Date: 13 March 2016

Quiz Dates:

Quiz 1: Week 6
Quiz 2: Week 11

Project Deliverables:

Requirements document; Increment 1: Week 5
Design document; Increment 2: Week 9
Test plan and test results; Increment 3: Week 12

7 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. As such, the accreditation criteria for the Software Engineering and Computer Science programs dictate that graduate attributes are taught and evaluated as part of the courses. This particular course aims at teaching and evaluating several graduate attributes. The following is a description of these attributes, along with a description of how these attributes are incorporated in the course.

(GA1) A Knowledge Base for Engineering: *Demonstrated competence in university level mathematics, natural sciences, engineering fundamentals, and specialized engineering knowledge appropriate to the program.*

The course will expand students ability to integrate and use specialized engineering knowledge to the program: Knowledge of software development process models, such as linear and iterative. Project management: roles, activities and deliverables for each software life cycle phase. Requirements management: analysis, elicitation, and scope. Architecture, design and the mapping of requirements to design and design to implementation. Traceability. Software quality assurance: verification, validation and the role of testing. Software maintenance and evolution.

(GA2) Problem analysis: *An ability to use appropriate knowledge and skills to identify, analyze, and solve complex engineering problems in order to reach substantiated conclusions.*

The project in this course is defined in such a way that requires the students to analyze the problem at hand before and determine for themselves exactly what needs to be done, and then determine how and with the help of what tools and software libraries it can be achieved.

(GA4) 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 project in this course is presented in an open-ended fashion, and its size and complexity is such that it needs to be tackled in teams. Students develop design artefacts such as architectural, and detailed design diagrams and documentation.

(GA5) Use of Engineering tools *is the ability to create, select, apply, adapt, and extend appropriate techniques, resources, and modern engineering tools to a range of engineering activities, from simple to complex, with an understanding of the associated limitations.*

The course teaches the use of the Java language, and leaves the students free to select what libraries that they will use in the project. Selection and use of the right tools and

libraries is a crucial aspect of accomplishing the practical work.

(GA6) Individual and Team Work: *An ability to work independently and as a member and leader in diverse teams and in multi-disciplinary settings.*

The course project is such that it needs to be tackled in teams. Students will cover a complete project development lifecycle, with project responsibilities rotating throughout the project. Project tasks will be assigned throughout the project to individual team members, which will require periodic integration of individual tasks into the common project context.

(GA7) Communication Skills: *An 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.*

Students will have to document their project progress throughout the project lifecycle by creating different types of software documents, such as requirements, design, and testing documentation following accepted standards.