SYSC 3020 – Introduction to SW Engineering  Spring/Summer 2017
Sherif Aly Ahmed
Email: sherifahmed@sce.carleton.ca

Undergraduate Calendar Course Description

SYSC 3020 [0.5 credit] Introduction to Software Engineering  Introduction to software engineering principles, software development life-cycles. Modelling in software engineering. Current techniques, notations, methods, processes and tools used in software engineering. UML modelling. Introduction to software quality, software verification and validation, software testing. Precludes additional credit for SYSC 3120 and SYSC 4120. Prerequisite(s): SYSC 2004 and SYSC 2006.

Book


Reference

Course Aims and Objectives

Software engineering is concerned with the theories, methods, and tools for developing complex, large-scale software. It encompasses a wide range of topics, including requirements elicitation and specification, software design, software construction (i.e., implementation), validation and verification, software maintenance, and the management of the software process. Every software development project involves one or more of these activities. With the Unified Modeling Language (UML) becoming the de-facto standard notation for software development in the IT industry, software development is becoming increasingly model-driven (or model-based), with less manual generation of source code and more automated generation of source code (from models).

A single course is clearly incapable of covering all these topics in depth. The aim of this course is to provide you with a broad understanding of the phases and activities in model-driven software development, and to introduce you to specific concepts that have not been covered systematically in first-year and second-year programming courses, yet are widely regarded as essential for engineering large software systems.

More specifically, the aims of this course are:

1. To understand how a software development life cycle consists of multiple phases, to
understand the role of each phase, the relationships between them, and the main principles that underlie these phases.
2. To learn model-based software development, using the UML to render the models.
3. To understand the challenges of software evolution.
4. To understand the challenges of software verification and validation.

The three main objectives/outcomes of this course are:

1. Students should be able to conduct requirement elicitation and produce software requirements in the form of use case models.
2. Students should be able to produce analysis models consisting of UML diagrams (class, sequence, state machine diagrams), following well-established heuristics.
3. Students should be able to conduct system design and object design using patterns.

Grades
- 25% Midterm Exam
- 45% Final Exam
- 20% Assignments
- 10% Labs

Office Hours
- TBD

Lab Periods

Attendance at the scheduled laboratory periods is mandatory, and attendance will be taken. During the labs you will work on short exercises that are intended to provide practical experience with tools and techniques related to the concepts presented in the lectures. You will normally be required to demonstrate and submit your lab work by the end of the lab period (or other specified deadline), as indicated in that week's lab "handout".

Your work in each lab period will be graded satisfactory, marginal, or unsatisfactory.

- Satisfactory means that you were present at the lab and made reasonable progress towards completing the lab exercises. Note that you do not have to finish all the exercises to receive a satisfactory grade. (75-100 percent).

- Marginal means that you made some progress towards completing the exercises, but your solutions to were not sufficiently complete to warrant a satisfactory grade. This grade indicates that you may be falling behind, and should take steps to remedy this situation. (50-75 percent).

- Unsatisfactory means that you were absent from the lab period, or you attended but made little or no progress towards completing the lab exercises. This indicates that you are likely having difficulty understanding important concepts and should seek help from your instructor as soon as possible. You will also receive

Partially adopted from previous outlines
unsatisfactory if you do not demonstrate or submit your work before the deadline or if it is apparent to the TA that you did not do enough of the lab work on your own; that is, you relied on your colleagues to explain the exercises and provide solutions. If you are absent from a lab period for any reason, it is up to you to do the missed lab work on your own time. Serious long-term illness will be dealt with on an individual basis. In these circumstances, please contact your instructor to discuss appropriate arrangements. (0-50 percent).

Assignments

Students are encouraged to discuss issues when working on assignments; however, you are expected to submit your own work for grading (assignments are individual work, unless otherwise stated). If you are unable to complete an assignment by the due date, you can submit the work you have completed.

Policy regarding late assignments: Any assignment submitted after the deadline, and up to 48 hours post the deadline will incur a penalty of ten percent of the assignment grade. No assignments are accepted after the 48 hour cutoff, unless with a documented excuse.

Exams

There will be one closed-book midterm test, which will be held approximately one-half of the way through the term. Computers will not be used during the midterm test.

Proposed Topics

- Introduction to Software Engineering. The nature of software; history and scope of software engineering; relationships with other fields; fundamental principles; software life cycles; overview of different software development processes.

- Requirement Elicitation using the UML. Producing a specification of the system that the client understands. Relationship between requirements and specifications; the uses of specifications, the qualities of specifications; the requirements engineering process and work products.

- Object Oriented Analysis using the UML. Producing an analysis model that the developers can unambiguously interpret. Formalizing the requirements (requirement elicitation) into specifications.

- System Design using the UML. Definition and objectives; object-oriented design with UML; architectural design; detailed design (with strong emphasis on design patterns); concurrent software; safety analysis and fault tolerance.

- Verification and Validation.