Carleton UNIVERSITY | Department of Systems and Computer Engineering

SYSC 3020
Introduction to Software Engineering

Calendar description

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.

Includes: Experiential Learning Activity.

Lectures three hours a week, laboratory three hours alternate weeks.

http://calendar.carleton.ca/undergrad/courses/SYSC/

Prerequisites

SYSC 2004 and (SYSC 2006 or SYSC 2002).

Precludes additional credit for SYSC 3100, SYSC 3120, SYSC 4120 and COMP 3004.

Prior knowledge

Students should:

- Have basic knowledge and understand fundamental concepts of Object Oriented programming.

Course objectives

Software engineering is concerned with the theories, methods, and tools for developing complex, largescale 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 goals of this course are:

- To understand how the 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.
- To learn model-based software development, using the UML to render the models.
- To understand the challenges of software evolution.
- To understand the challenges of software verification and validation.

List of topics

- Introduction to Software Engineering. The nature of software, history and scope of software engineering, relationships with other fields, fundamental principles, software life cycle.
- Requirement Elicitation. Using 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 products.
- Object Oriented Analysis using UML. Producing an analysis model that the developers can unambiguously interpret. Formalizing the requirements (requirement elicitation) into specifications (Analysis).
- System Design. Definition and objectives, object-oriented design with UML, architectural design, detailed design, concurrent software, safety analysis and fault tolerance.
- Design Patterns, Verification, and Validation.
- Software Testing.

Learning outcomes

By the end of this course, students should be able to:

- Conduct requirement elicitation and produce software requirements in the form of use case models.
- Produce analysis models consisting of UML models (composed of class, sequence, state machine diagrams), following well-established heuristics.
- Conduct system design and object design using patterns.

Graduate Attributes (GAs)

The Canadian Engineering Accreditation Board requires graduates of engineering programs to possess 12 attributes at the time of graduation. Activities related to the learning outcomes listed above are measured throughout the course and are part of the department’s continual improvement process. Graduate attribute measurements will not

be taken into consideration in determining a student's grade in the course. For more information, please visit: https://engineerscanada.ca/.

Graduate Attribute	Learning outcome(s)
1.8.S: Knowledge Base: Developed: Software Engineering	1, 3
2.1: Problem Analysis: Developed: Problem definition	1
2.2: Problem Analysis: Introduced: Approach to the problem	1, 2
2.3: Problem Analysis: Introduced: Use of assumptions	1
4.1: Design: Developed: Clear design goals	2, 3
4.4: Design: Developed: Design solution(s)	3
5.1: Use of Engineering Tools: Developed: Diagrams and engineering sketches	2

Accreditation Units (AUs)

For more information about Accreditation Units, please visit: https://engineerscanada.ca/.

The course has a total of 46 AUs, divided into:

- Engineering Science: 100%

Instructor and TA contact

Specific to course offering (tbd)

Textbook (or other resources)

Specific to course offering (tbd)

Evaluation and grading scheme

Specific to course offering (tbd)

Breakdown of course requirements

Specific to course offering (tbd)

Tentative week-by-week breakdown

Specific to course offering (tbd)

General regulations

Specific to course offering (tbd)