Carleton UNIVERSITY | Department of Systems and Computer Engineering

SYSC 3120
Software Requirements Engineering

Calendar description

Current techniques, notations, methods, processes and tools used in Requirements Engineering. Requirements elicitation, negotiation, modeling requirements, management, validation. Skills needed for Requirements Engineering and the many disciplines on which it draws. Requirements analysis: domain modeling, modeling object interactions; UML modeling. Introduction to software development processes.

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 enrolment in Software Engineering.

Precludes additional credit for SYSC 3020 and COMP 3004.

Prior knowledge

Students should have knowledge of:

- Software development concepts.
- Discrete mathematics and logic (propositions, predicates, etc.).

Course objectives

Requirements engineering is a branch of software engineering concerned with discovering, analyzing, modeling, validating, testing, and writing the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time, and across software families.

As more and more features are being defined for software systems, requirements engineering is becoming increasingly important to help manage the development and evolution of features throughout the product lifecycle. Requirements engineering has proven to be one of the most difficult and critical activities for the successful development of software and software-intensive systems. Contrary to common belief, software rarely fails. More often than not, software behaves exactly as it was designed to, but it was the requirements that were flawed. Some sources assert that over 90% of software issues result from deficient requirements. For example, if requirements are

missing from a specification, then even the most careful implementation of a system will not result in a product that is complete or useful. Moreover, if requirements are included in the specification that are not actually valid, then the product or system becomes unnecessarily expensive. Therefore, experience teaches us that getting requirements right, and precisely specifying them, is essential for the establishment of safe, secure, and effective systems. As a result, solid requirements engineering has been recognized as the key to improved, on-time, and on-budget delivery of software and systems projects.

This course examines current software requirements engineering techniques, notations, methods, processes, and tools. It will cover informal, semi-formal, and formal approaches, while striking a balance between theory and practice. This course will involve building models of both software engineering processes and products, concerning both functional and non-functional goals, requirements, and specifications. It will provide students with the skills needed for software requirements engineering, as well as a foundation that can be used to systematically establish, define, and manage the requirements for large, complex, and changing software-intensive systems, from technical, organizational, and management perspectives.

List of topics

- Introduction to Requirements Engineering:
    - Software Engineering Processes
    - Definition, Role, and Importance of Requirements Engineering
    - Requirements Engineering Activities and Processes
    - Types of Requirements (Functional vs. Non-Functional)
    - Techniques for Writing High-Quality Requirements
- Requirements Inception, Elicitation, Evaluation, and Documentation:
    - Domain Understanding, Product Vision, and Scope
    - Elicitation Approaches and Techniques
    - Business Events, Viewpoints, and Stakeholders
    - Requirements Evaluation and Risk Management
    - Software Requirements Specification (SRS) Documentation Standards and Templates
- Requirements Analysis, Modeling, and Specification:
    - Description vs. Specification
    - Modeling Techniques and Diagrammatic Notations
    - Use Case Modeling and Scenario Descriptions
    - Analysis for Understanding the Domain and Requirements
    - Formal Specification
- Requirements Verification and Validation:
    - Verification vs. Validation
    - Techniques for Inspection, Verification, and Validation
    - Detection of Conflicts, Inconsistencies, and Completeness
- Requirements Evolution and Management:
    - Requirements Traceability
    - Priorities, Changes, Baselines

    - Reusing Requirements
- Advanced Topics:
    - Hazard Analysis and Safety-Critical System Requirements
    - Threat Modeling and Security-Critical System Requirements
    - Tool Support for Requirements Engineering
    - Open Research Problems in Requirements Engineering

Learning outcomes

By the end of this course, students should know and understand:

- The role of, and need for, requirements engineering for the development of software systems.
- The breadth of skills needed for requirements engineering, and the many disciplines upon which it relies.
- The main principles that underlie requirements elicitation, evaluation, documentation, and quality assurance.
- How to elicit and document high-quality requirements.
- How to effectively use visual models, diagrams and notations in requirements analysis.
- Critical issues affecting the success of various approaches in requirements engineering.

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

- Explain and identify the different types of requirements.
- Conduct requirements elicitation to describe the visible behaviour of a software system.
- Conduct requirements evaluation activities to make decisions based on multiple criteria.
- Specify and analyze scenarios using consistent diagrammatic (or graphical) notations and/or appropriate formal methods.
- Prepare well-organized, and maintainable software requirements documentation that can be reviewed, corrected, and (eventually) accepted by clients and stakeholders.
- Describe and participate in requirements verification, validation, and traceability activities.

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	1-7
2.1: Problem Analysis: Developed: Problem definition	8
2.2: Problem Analysis: Developed: Approach to the problem	10, 11, 12
3.1: Investigation: Developed: Complex problem assessment	9
5.1: Use of Engineering Tools: Developed: Diagrams and engineering sketches	10

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)