Carleton UNIVERSITY | Department of Systems and Computer Engineering

SYSC 4101
Software Validation

Calendar description

Techniques for the systematic testing of software systems. Software validation and verification, software debugging, quality assurance, measurement and prediction of software reliability. Emphasis on the treatment of these topics in the context of real-time and distributed systems.

Includes: Experiential Learning Activity.

Lectures three hours a week, laboratory/problem analysis three hours a week.

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

Prerequisites

SYSC 3120 or SYSC 3020.

Precludes additional credit for COMP 4004.

Prior knowledge

Students should:

- Be able to program with a procedural programming language as well as an object-oriented programming language.
- Be able to trace the execution of such programs, that is understand the flow of control of programs.
- Be able to understand a high-level design, a specification with a (extended) finite state machine.
- Be knowledgeable in discrete mathematics.

Course objectives

Software Verification and Validation (V&V) are two important activities of any software development. One of the main techniques used for V&V is called software testing.

Some data reported in literature indicate that software testing usually amounts for 30% to 40% of the total software development cost, and that for safety critical software this percentage can go up to 70%. Other anecdotal evidence of the importance of software testing is that in large projects, the amount of test code (for instance measured in number of lines of code) can be double the amount of application code! Although a lot of testing is conducted, there are still many defects in released software (verification issue) and software do not entirely satisfy customer needs (validation issue).

One of the main limits of today's testing activities is that they are often not conducted in a systematic, repeatable way, using clear rationale. For instance, a study reported that open source software development projects lack "attention to basic, accepted, and mature testing techniques."

The main purpose of SYSC-4101 is to introduce you to basic, well-accepted, systematic, mature software testing techniques so you become capable of using them wisely; to introduce you to the notions of validation, verification and testing so you can precisely understand what activity you conduct in practice; to introduce you to the challenges of using testing techniques and design software test harness.

List of topics

- Context and Definitions
- Category-Partition software testing
- Graph criteria
- Control flow graph (application of graphs)
- State based testing (application of graphs)
- Inheritance/Generalization and testing
- Integration testing
- Criteria for logic expressions
- Regression testing
- Drivers, stubs and oracles
- Other considerations, including empirical software engineering

Learning outcomes

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

- Describe and discuss the concepts of verification, validation, testing, test model, test criteria.
- Describe the benefits and limitations of software testing.
- Understand results of a testing campaign in relation to the notion of faults, errors and failures, to the test scaffolding.
- Apply testing techniques for unit testing, integration testing, and system testing
- Apply standard black-box and white-box testing techniques.
- Discuss advantages and drawbacks of standard black-box and white-box testing techniques.
- Describe the problem of regression testing.
- Describe problems specific to procedural, object-oriented, distributed, or real-time software.
- Describe the general notion of quality assurance.
- Implement (design) test suites for standard black-box and white-box testing techniques.
- Appraise alternative testing techniques, while accounting for their advantages and drawbacks, for specific software development contexts.

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: Applied: Software Engineering	1-10
2.4: Problem Analysis: Developed: Interpreting the solution - validity of results	2
3.5: Investigation: Applied: Interpretation of data (synthesis) and discussion	3
5.1: Use of Engineering Tools: Applied: Diagrams and engineering sketches	5

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: 50%
- Engineering Design: 50%

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)