SENG 265: Software Development Methods

Fall 2025

Territory Acknowledgement

We acknowledge and respect the lək̓ʷəŋən peoples on whose traditional territory the university stands, and the Songhees, Esquimalt and WSÁNEĆ peoples whose historical relationships with the land continue to this day.

Who am I?

- Roberto Almeida Bittencourt
  - Born in Brazil
  - Electrical Engineer (Brazil, 1996), Master in Computer Engineering (Sweden, 2000), and PhD in Computer Science (Brazil, 2012)
  - Previously in Canada for two years, at UBC in Vancouver, from 2010 to 2012
  - Have worked at the State University of Feira de Santana from 2000 to 2023
  - Have worked at the University of Victoria since 2023 as an Assistant Teaching Professor
- I enjoy teaching courses on Introductory Programming and Software Engineering

SENG 265

- **Software Development Methods**
- Instructor: Roberto A. Bittencourt
  - e-mail: rbittencourt@uvic.ca
    (please include "SENG 265" in subject line)
  - Office: ECS 552
- Labs:
  - Begin week of September 8
  - Will take place in person (ELW B238)
- Lab Instructors and Graders:
  - Ashiq Ullah (ashiqullahmg@uvic.ca);
  - Jonah Carroll Dullaert (jcarrolldullaert@uvic.ca);
  - Mostafa Abdollahi (abdolahi68@uvic.ca).

Administrative Details

- Office hours:
  - Tuesdays, 3:30pm - 4:20pm at ECS 552
  - Thursdays, 2:30pm - 3:20pm at ECS 552
  - Fridays, 1:00pm - 2:20pm at ECS 253 (group room)
- Course website:
  - Via BrightSpace
  - For lecture slides, some lab material, assignments, announcements, and discussion forums (send your questions there, they may be useful to everyone)
  - Link to course outline can be found at Brightspace site for our course.

Labs

- 10 labs
- Lab sections:
  - Our focus is on hands-on + tutorial components
  - You must register for a lab section
  - Attending labs (i.e., doing the lab exercises at the day/time of the lab section you enrolled in) is required for assessment
  - Lab 1 will be done individually
  - The remaining labs will be done in pairs
    - Good to get used to pair programming (please alternate roles)
- Lab grading:
  - 8 best grades out of 10 labs
    - Concessions are handled uniformly for all students by taking the 8 best grades out of 10 labs
  - 10% weight in the final grade

Assignments

- 5 assignments
  - 2 individual assignments
    - software development basics in C
    - dynamic memory in C
  - 3 pair assignments in Python (developing a full application)
    - Model
    - Persistence
    - User Interface
- Assignment grading:
  - 4 best grades out of 5 assignments
    - Concessions are handled uniformly for all students by taking the 4 best grades out of 5 assignments
  - 45% weight in the final grade

Grading

- Breakdown:
  - Assignments: 45% (5 assignments, same weight)
    - **Must obtain a passing grade (50%) in the average of all the assignments**
    - Worst assignment grade will be dropped
  - Lab work: 10% (10 labs)
    - 2 worst lab grades will be dropped
  - One in-class midterm: 15%
    - **October 15 (Wednesday), 6pm, at ECS 123**
  - Final exam: 30%
    - **Must obtain a passing grade (50%) in the final exam**
    - scheduled by the university in the final exam period
- Marking concerns (“one-week rule”)

Purpose of Course

- General introduction to:
  - UNIX/Linux environment and scripting
  - production languages (C & Python)
  - software development practices
- Introduction to software engineering concepts and vocabulary
- Working at a higher level of abstraction
- Acquiring and reinforcing good habits when writing software and software systems

Context

- Your experience thus far:
  - small, relatively simple programs
  - provided with steps to solving specific problem
  - written alone
  - no ongoing maintenance
- What awaits in industry:
  - large and complex projects
  - do not know ahead of time how to solve the problem
  - work in teams (often very specialized)
  - ongoing maintenance is critical

Course topics

- UNIX/Linux fundamentals
- C programming
- Python programming
- Source code control, code revision and change management
- Inspection, testing and debugging
- Introduction to software development practices and related concepts

By the end of this course…

- You should be able to:
  - program with some comfort in a UNIX environment
  - use Python for prototyping, and other kinds of scripting for support of code testing and debugging
  - recognize a problem statement that can become a program specification
  - use general-purpose languages such as C and Python to solve programming problems
  - work with code versioning systems (git) to manage changes in your own code
  - apply some general software engineering techniques to your own projects
  - be ready to delve deeper into more formal software engineering approaches

Academic integrity

- Guiding principles
  - discussion is encouraged …
  - .. but work submitted for credit must be your own
  - in cases where attribution is appropriate, it must be given
    - example: code taken from a textbook or web-based tutorial
    - example: algorithm based on a journal paper
- If you are unclear, please ask the instructor.
- Attribution for the course slides!
  - Various were originally created by Mike Zastre, Nigel Horspool and some other instructors, with some changes by me. I created the slides on models and some on persistence with files.

Development environment (1)

- At first glance there would appear to be two families of development tools:
  - those which employ a **GUI** (graphical user interface), typically as part of an **IDE** (integrated development environment)
  - those which employ a **CLI** (command line interface)
- An IDE tends to hide many of the details of the development process

Development environment (2)

- In the first part of this course, we will use a command-line interface;
  - this gives us a better understanding of aspects of the development process which might be hidden inside an IDE
  - compilation
  - source code management
  - profiling
  - testing, etc.
- Our command-line interface ("bash") is run within the Linux variant of Unix.
- Command-line code editors
  - vi / vim (use it only if you learn and practice all of its commands)
  - nano (easier to use and navigate than vi)
  - gedit (command-line editor within the Linux GUI)

Development environment (3)

- In the second part of this course, we will use a graphical user interface;
  - this gives us more productivity
  - integrated environment
  - integration with plugins
- You may still use git from the command line
  - Trust me, this is rather common
- Or you may use a git plugin from your favorite IDE (VS Code, IntelliJ IDEA, etc.)
- Some testing scripts will be run from the command line (helps with testing automation)

Development environment (4)

- Local access:
  - Just go to the lab at ELW 238
- Remote access:
  - You will need **VPN** access
  - Use your **ssh** client
  - Log in to **ugls.ece.uvic.ca**
- Test your code in a lab machine
  - Even if you code your solution first in your laptop
  - We will grade your code by running it in the lab

One word about Gen AI

- Gen AI is getting more popular as tool to support developer work, but...
- Students need to master software development skills whose learning requires tackling complex problems without Gen AI
- In SENG 265, Gen AI can be used as a comparison after you master the skills with practical work