CSC301

Introduction to Software Engineering

Winter 2018

What is CSC301 about?

The practice of making software products

- In a realistic environment
- With users in mind
- With a clear purpose and value
- On an ongoing basis, in a traceable manner
- As part of a team
- While dealing with changing requirements

Course Goals

- Improve coding skills
- Introduce software development life-cycle (SDLC)
  - The tasks, tools, practices and conventions used by software professionals when creating and maintaining software products.
  - Much more than just coding!
- Get you to think as pragmatic professionals
  - Tooling as part of your workflow
  - Articulate goals, define success metrics and make data-driven decisions
  - Identify common problems/challenges and apply well-known, generic solutions

Topics

- Software Tools
  - Version control (Git)
  - Project management (GitHub)
  - Build and/or automation (Travis CI, Maven)
  - IDE and/or debugger (IntelliJ, Eclipse)
- A pragmatic approach to tooling
  - Professionals use tools to be more efficient
  - And build custom tools (focusing on “bang for the buck”), when they are needed.
  - Goal: Maximize productivity

Topics

- Project management
  - Software processes
  - Focus on modern Agile techniques
  - E.g.: Scrum, Kanban, and Test-Driven Development (TDD)
- A pragmatic approach to team organization
  - Collaboration comes with an overhead
  - Organize a team’s workflow ⟺ Reduce overhead
  - Goal: Minimize overhead (i.e. maximize productivity)

Topics

- Product management
  - Articulating *what* we're building, *who* we're building it for and *why* it is useful/valuable
  - Standard planning tools/techniques such as personas, user stories, diagrams, mock-ups, etc.
  - Scoping and defining a Minimum Viable Product (MVP)
- A pragmatic approach to product decisions
  - Define success metrics ➔ Experiment & collect user feedback ➔ Make data-driven decision(s)
  - Do it frequently and incrementally improve your product
  - Goal: Maximize benefit/utility/value

Topics

- Software design & Coding
  - Best practices and common pitfalls
  - Design patterns such as Iterator, Adapter, DAO, Observer/Observable, Abstract Factory, and Builder
  - Code craftsmanship
  - Various useful topics in software engineering
    - E.g.: Lambda expressions (aka callbacks), serialization and persistence, asynchronous programming, lazy-loading and caching, distributed applications, etc.

Course Logistics ...

Resources

- Course website:
  https://csc301-winter-2018.github.io/
- Discussion Board:
  https://piazza.com/utoronto.ca/winter2018/csc301/home
- GitHub organization:
  https://github.com/csc301-winter-2018

Instructor - Evening Section

- Alexei Lapouchnian
    - Email: alexei.lapouchnian@utoronto.ca
        - Begin email subject lines with “[CSC301]”
        - If your question is of general interest to the class, please consider posting it on the discussion board (Piazza), instead of sending an email
    - Office hours: **Tuesday 16:30-17:30** in **BA3219**

Instructor - Day Section

- David Jorjani
    - Email: jorjani@cs.toronto.edu
        - Begin email subject lines with “[CSC301]”
        - If your question is of general interest to the class, please consider posting it on the discussion board (Piazza), instead of sending an email
    - Office hours: **Tuesday 18:45-19:45** in **BA3219**

Head TA

- Adam El-Masri
  - Email: adam.el.masri@mail.utoronto.ca
    - Begin email subject lines with “[CSC301]”
  - Responsible for:
    - GitHub infrastructure
    - Assignment deployments and automarking
    - Certain lectures

Lectures & Tutorials

- Day Section, L0101
  - Lecture: Tuesday 12:00-14:00 @ GB303
  - Tutorial: Thursday 13:00-14:00 @ GB303, LM155, and BA2185
- Evening Section, L5101
  - Lecture: Monday 18:00-20:00 @ MP137
  - Tutorial: Monday 20:00-21:00 @ MP137, BA1200, and BA1210

Prerequisites

- CSC209 - Software Tools and Systems Programming
    - Implicit prerequisite, CSC207 - Software Design
    - Basic Object-Oriented programming in Java
    - Comfortable with Unix command line
- CSC263/CSC265 - (Enriched) Data Structures and Analysis
    - Understand the difference between *data type* (interface) and *data structure* (implementation)
    - Basic data structures and types
    - E.g.: Array, List, Queue, Stack, Map (aka dictionary), Tree, Graph

Marking Scheme

4 Individual Programming Assignments	30%
Term Test	25%
Team Project (3 deliverables throughout the term)	45%

No Final exam!

Individual Assignments

- 4 Java coding assignments
- Auto-marked
- Focus:
    - Reading and writing object-oriented code
    - Hands on experience with professional tools
      E.g.: Git, GitHub, Travis CI and Maven
    - Applying design patterns
- Meant for you to get 100%
    - The task is clearly specified (as JUnit tests)
    - You can submit as many times as you want
    - Travis CI is used for verifying your submission

Individual Assignments

- Auto-marked assignments ⇒ Strict deadlines & No exceptions
  - Auto-marker rolls back changes that were committed after the deadline
  - It is your responsibility to make sure your code compiles!

- Start early and avoid last-minute, *unexpected* technical issues
  - If your first commit is from the last 24 hours before the deadline, you are taking full responsibility for any unexpected issue that may occur.
  - Responsible professionals prepare for unexpected issues, and so should you

Term Test

- Two topics:
  - Git/GitHub
  - Applying design patterns to solve common engineering problem
- Tests your ability to communicate (i.e., read/write) using code
  - Focus is on software design, *not* algorithms
- Based on the individual assignments
  - Therefore, if you don't understand something about the assignment, you should ask
    - During office hours, on the discussion board or in class
- Meant to be fairly challenging
  - There are no easy questions

Team Project

- ~8 weeks long
- 6-7 students per team
- One TA per team, acting as a “mentor”
- Focus:
  - Identifying users and need
  - Defining a product
  - Building a prototype/MVP
  - Organizing a team
  - Working in a traceable manner
  - Presenting your work

Team Project

- 25% - Three deliverables
  - Concise deliverables presenting your work
  - Meant to be useful, not to add extra work
  - Evaluated by the TAs
- 10% - Final demo
  - During the last week(s) of the term
  - Evaluated by the instructor(s)
- 8% - Consistent individual contribution
  - Commit history & graphs on GitHub
  - You are expected to contribute valuable work, at the very least, twice a week
- 2% - Tutorial participation
  - Participation in tutorials throughout the term (0.25% for every tutorial attended, up to 2%)

Cheating

- Don’t cheat!
- Feel free to discuss ideas with others, but don’t take notes or share code with others
- When in doubt, ask your instructor or TA

Please keep in mind that CSC301 is a hands-on course!

In other words - A lot of fun, but also a lot of work.