Michael Adams

Department of Electrical and Computer Engineering
University of Victoria
Victoria, BC, Canada
https://www.ece.uvic.ca/~mdadams

Summer 2023
Section 1.1

Course Project
The information about the course project presented on these slides is *not intended as a replacement for the project handout*. These slides are only intended to supplement the information on the project handout. Some very important details about the project are only presented on the project handout (and not on these slides). So, it is extremely important to read the project handout carefully (in addition to looking at these slides).
Project

- develop some significant software project using C++ programming language
- SENG475 students can work either individual or in teams
- size of team considered in grading of project
- ECE596C students must work individually
- stages of project work (in order):
  1. preliminary project discussion
  2. formal written proposal
  3. project software, including brief video presentation
- some video presentations for past projects available via links on course web site
- past projects only for illustrative purposes, not for simply copying ideas
- since project software due very late in term, penalty for late project software submissions quite severe (grows exponentially with time)
- some students use project as way to market themselves to prospective employers
Skillsets Developed by Project

- designing (as opposed to implementing) software
- designing software interfaces; for example:
  - command-line interface (CLI)
  - text-based user interface (TUI)
  - graphical user interface (GUI)
  - application programming interface (API)
- writing formal specifications of software interfaces
- software project planning
- effective testing
- written communication skills
- oral presentation skills
- further develop proficiency in C++
- each student required to meet with instructor to introduce idea for project
- instructor will assess whether idea has potential to lead to viable project
- student should only to proceed to prepare written proposal if instructor has indicated that project idea seems viable
- preliminary discussion intended to prevent student from spending time preparing proposal for project that is very unlikely or guaranteed not to be approved
- project idea could be rejected for numerous reasons, such as:
  - another student already doing similar project
  - idea would lead to project with too much or too little work
  - idea not of sufficient relevance to course
- meetings normally take place during office hours or free time during lab/tutorial time slots
student required to prepare formal written proposal describing project software to be developed

proposal must provide *clear, concise, and complete* description of project to be undertaken

should be as short as possible subject to constraint that *all required information must be included*

project handout provides very detailed information about what information must be included in proposal

student forbidden from starting work on project software until proposal approved by instructor (in order to prevent work being done on project that would not be appropriate for course)

proposal has no submission deadline per se, but if project not approved at least three weeks before end of lectures, student would be at grave risk of failing project (due to insufficient time to complete it)
-grade for proposal assigned based on first version submitted, regardless of whether proposal approved

-if proposal not approved, need to resubmit revised version

-if revised version must be submitted, *must use highlighting or change bars* (or something similar) to clearly indicate directly on the document itself what text has been changed so instructor only needs to re-read parts that have changed

-if number of revisions of proposal becomes excessive, instructor reserves right to deduct further marks from proposal

-grading approach for proposal intended to encourage good job *on first version submitted*, eliminating need for revision
Project Software

- project software submitted via GitHub Classroom
- particular Git repository layout must be used
- software must build and run on lab machines using SDE for course
- only libraries installed on system (as part of base OS install or SDE) can be used (i.e., students cannot include other libraries as part of their project software)
- project software submission required to include URL for video presentation
- submission deadline for project software during final exam period (with specific date posted on course web site)
- if early submission deadline met, entire project grade (i.e., proposal and software) scaled by multiplicative factor of 1.05
- must notify instructor by email when project software considered submitted for grading
- student required to prepare brief video presentation for project
- presentation must not exceed 5 minutes
- generally introduce project using some slides
- give brief demonstration of software developed
- be sure to include student name and project title on title slide
- video must be hosted by YouTube
- use either unlisted or public visibility (do not use private visibility)
Common Problems With Project Proposal I

1. extremely vague project specification
   - how many software components being delivered and of what type (e.g., application program, library, etc.)?
   - what do each of software components do and how are they used?

2. what part of software to be developed by student not specified or ambiguous
   - almost all code relies on libraries to some extent
   - extremely important what part of functionality in project software provided by code written by student versus code in libraries
   - has massive impact on feasibility of project and whether project is even appropriate in first place

3. completely missing or incomplete/ambiguous specification of user interfaces
   - instructor will build, run, and test software
   - instructor cannot run software unless user interface (e.g., CLI, TUI, GUI) completely and unambiguously specified
Common Problems With Project Proposal II

- Instructor cannot assess suitability of proposed project unless interfaces clearly understood since interfaces can very significantly impact amount of work involved

4 CLI not fully and clearly specified
- All command-line arguments and options must be fully documented
- Instructor cannot test software if cannot understand clearly how to run programs that are part of project

5 Custom file formats used by software not fully and clearly specified
- Instructor cannot generate datasets for testing software if formats not clearly and unambiguously specified

6 If software has GUI, need some rough sketches to give general idea of how interface will work

7 No evidence that student has ensured that project is feasible; for example, failure to identify
  - What parts of what libraries will be used
  - What methods will be used
Common Problems With Project Proposal III

8. failure to check that library to be employed provides functionality required by project

9. failure to check that library to be employed (including header files) installed on lab machines and version of library is sufficiently new to provide all required features

10. project is not testable
   - software must be defined in manner that make easy testing possible
   - problematic if software computes solution to problem for which there is no practical way to check if results appear reasonable
   - software that consists only of library is not feasible to test (since instructor would have to learn library and write application programs to test library)

11. poor quality writing
   - writing is unclear/ambiguous, poorly organized, overly verbose, etc.

12. poor choices made in design of user interface
   - using interface that constraints users in arbitrary ways for no justifiable reason
- using many cryptic single-letter CLI options (instead of using long option names)  

- not using standard input, standard output, and standard error appropriately  
  - if program has single data stream as input and produces single data stream as output, unreasonable to read input from file and write output to file  
  - use standard input and standard output  

- failure to consider instructor time constraints  
  - reviewing project proposals takes significant time  
  - avoid situations that would lead to many students wanting to meet with instructor in short time window of several days
Good Strategy for Quality Proposal

- swap proposals with another student
- other student must have no knowledge of project
- ask them to read proposal and then explain exactly how they would:
  - generate input datasets
  - specify command line options
  - navigate through GUI if software has GUI
  - examine output for correctness
- if other student cannot do these things, this is indication of deficiencies in proposal
- given collection of numbers, sorts numbers, and outputs sorting result
- appropriate for project?
- feasible for project?
what exactly is meant by “numbers”?
what is interface with user? GUI? TUI? CLI?
what is sorting criterion?
is more than one sorting algorithm supported?
what sorting algorithm is used?
where does input originate?
how exactly is data formatted?
where is output sent?
how exactly is output formatted?
what aspects of program behavior can be controlled?
are sorting methods implemented by directly by project software or library?
project software consists of single application program called `sort`

specification for CLI might include information similar to output of “man sort”

for example, see: