

Computing and Software 741 (Computational Engineering and Science 741) Development of Scientific Computing Software

September 4, 2019

This course outline contains important information that will effect your grade. You should retain and refer to this outline throughout the term.

1 Instructor

Dr. Spencer Smith

Office: ITB/167

E-mail: smiths@mcmaster.ca

Web: <http://www.cas.mcmaster.ca/~smiths>

Office Hours: Drop in, or by appointment

2 Calendar Description

This course presents the basic principles of software development for reliable and sustainable scientific and engineering software. Using example applications, a systematic process is given for the development and documentation of requirements, system design, module design, implementation, testing and inspection.

3 Introduction

Scientific computation consists of using computer tools to simulate mathematical models of real world systems so that we can better understand and predict the system's behaviour. A small sample of some important applications of scientific computation include the following:

designing new automotive parts, analysing the flow of blood in the body, and determining the concentration of a pollutant released into the groundwater. As these examples illustrate, scientific computation can be used for tackling problems that impact such areas as manufacturing, financial planning, environmental policy, and the health, welfare and safety of communities. Given the important applications of scientific computation, it is surprising that little emphasis is currently placed on the quality of the software that performs the computations. Although many successful and sophisticated algorithms have been developed for scientific computing, the software often suffers from problems with such qualities as reliability, usability, verifiability, maintainability, reusability and portability. This is why scientific software is routinely sold with a disclaimer instead of a warranty. To make matters worse, the quality of scientific software is becoming increasingly more of an issue because the complexity and size of the problems that can be simulated on modern computers is constantly growing. The question for the future is how to meet the growing need for providing quick solutions to large and complex problems, and at the same time ensure that the solutions are correct? This graduate course will investigate this question by applying to scientific computing problems such software engineering methodologies as commonality analysis, requirements analysis and documentation, modular decomposition, module interface specification, testing, code and document generation and assurance cases.

The course will look at tools, techniques and principles for iterative and incremental development of scientific and engineering software. Despite the iterative development cycle, the documentation will follow 5 rational steps: i) identify the problem, ii) document the requirements, iii) design the system, iv) implement the software, and v) perform tests. This structure is well suited to scientific computing because it parallels the idealized scientific method, as follows: i) a physical problem of engineering or scientific importance is identified; ii) a system of governing equations and the associated boundary conditions are derived; iii) a numerical algorithms are developed; iv) the numerical algorithms are implemented on a computer; and, v) the model and the computed results are verified and validated, with the potential to return to one of the previous steps if necessary. These five steps are inherently multidisciplinary as they involve skills from physical modelling, mathematics, numerical analysis and computer science. For this reason it is important that requirements (including assumptions) and design decisions are clearly documented.

Course Web Site

This course will be administered via Avenue to Learn. Go to

<http://avenue.mcmaster.ca/>

to access the course's Avenue to Learn page. Please send only normal McMaster e-mail; do not send mail via Avenue.

Students should be aware that, when they access the electronic components of this course, private information such as first and last names, user names for the McMaster e-mail accounts, and program affiliation may become apparent to all other students in the same course. The available information is dependent on the technology used. Continuation in this course will be deemed consent to this disclosure. If you have any questions or concerns about such disclosure please discuss this with the Instructor.

It is the student's responsibility to be aware of the information on the course's Avenue to Learn page and to check regularly for announcements.

The primary purpose of Avenue will be for maintaining grades. Most of the course content will be maintained in a public git repository. You can access this repository at:

<https://gitlab.cas.mcmaster.ca/smiths/cas741/>

Rather than use the Avenue discussion board, please post your questions (issues) to the GitLab issue tracker.

In addition to Avenue and the Gitlab course note repository, every student will create a public gitHub repository (with the instructor added as a full access collaborator) for their work. The GitHub server is located at <https://github.com/>. Students will be expected to use GitHub to provide comments on the work of other students in the class.

4 Course Project

At the beginning of the term each student will select a scientific computing problem. Over the course of the term software will be developed to address the selected problem. The software development process will follow the iterative waterfall model, with the following milestones:

1. Software Requirements Specification (SRS)
2. Module Guide (MG)
3. Module Interface Specification (MIS)
4. Implementation (any appropriate programming language)
5. Verification and Validation (VnV) Plan (divided between system and unit documents)
6. Verification and Validation (VnV) Report (divided between system and unit documents)

With approval from the instructor, the deliverables can potentially be modified, if a project is more suited to a different structure. For instance, a project could replace one of the above deliverables with an assurance case deliverable, or with domain specific code to automatically build the deliverables.

5 Course Structure

The format of the course will consist of student and instructor presentations. Each student will be expected to do an informal presentation on some subset of their SRS, MG, MIS, Implementation, VnV Plan and VnV Report. It is expected the class discussion will assist in improving the quality of the written deliverables. Each student will be expected to hand in the following written documents: SRS, MG, MIS, VnV Plan, VnV Report, code and Final Documentation.

6 Grading

1. Presentations and class discussion 5%
2. “Domain Expert” and secondary reviewer roles 10%
3. Problem Statement 0%
4. System Requirements Specification (SRS) 15%
5. System Verification and Validation (VnV-Syst) Plan 15%
6. Module Guide and Module Interface Specification (MG and MIS) 15%
7. Final Documentation 40%
 - (a) Problem Statement (Revised)
 - (b) SRS (Revised)
 - (c) System VnV Plan (Revised)
 - (d) MG (Revised)
 - (e) MIS (Revised)
 - (f) Unit VnV Plan
 - (g) Code
 - (h) System VnV Report
 - (i) Unit VnV Report

8. Drasil simple physics example, pull request accepted 5% (Bonus)

Each student will be assigned as a “Domain Expert” (DE) for one of their classmates. The DE will review (via GitHub issues) their assigned project. In addition, they will write the code for one of the modules of their assigned project and issue a pull request through GitHub. For the secondary reviewer role, the instructor will assign various review tasks throughout the term. The secondary reviewers feedback will focus more on the structure and format of the documentation, rather than on issues related to domain knowledge.

The potential bonus marks involve creating a simple physics example in Drasil (<https://github.com/JacquesCarette/Drasil>). Drasil represents recent (still incomplete) work to automate many of the tasks that are currently done manually for design and documenting SC software.

7 Policy Statements

This section of the course outline explains the course policy with respect to improving the course, missed work, discrimination and academic dishonesty.

7.1 Improving the course

Ideas to improve the course are always welcome. Moreover, if you have problems in the course please contact the instructors as early as possible.

7.2 Missed Work

Late assignments, without permission, will receive a penalty of -20 % per working day that the assignment is late.

7.3 Discrimination

The Faculty of Engineering is concerned with ensuring an environment that is free of all adverse discrimination. If there is a problem, that cannot be resolved by discussion among the persons concerned, individuals are reminded that they should contact their Department Chair and the Human Rights and Equity Services (HRES) office as soon as possible.

7.4 Academic Dishonesty

You are expected to exhibit honesty and use ethical behaviour in all aspects of the learning process. Academic credentials you earn are rooted in principles of honesty and academic integrity.

Academic dishonesty is to knowingly act or fail to act in a way that results or could result in unearned academic credit or advantage. This behaviour can result in serious consequences, e.g., the grade of zero on an assignment, loss of credit with a notation on the transcript (notation reads: “Grade of F assigned for academic dishonesty”), and/or suspension or expulsion from the university.

It is your responsibility to understand what constitutes academic dishonesty. For information on the various types of academic dishonesty please refer to the Academic Integrity Policy, located at

<http://www.mcmaster.ca/academicintegrity/>

The following illustrates only three forms of academic dishonesty:

1. Plagiarism, e.g., the submission of work that is not one’s own or for which other credit has been obtained.
2. Improper collaboration in group work.
3. Copying or using unauthorized aids in tests and examinations.

Your work must be your own. Plagiarism and copying will not be tolerated! If it is discovered that you plagiarized or copied, it will be considered as academic dishonesty.

Students may be asked to defend their written work orally.

Course Modifications

The instructor and university reserve the right to modify elements of the course during the term. The university may change the dates and deadlines for any or all courses in extreme circumstances. If either type of modification becomes necessary, reasonable notice and communication with the students will be given with explanation and the opportunity to comment on changes. It is the responsibility of the student to check their McMaster e-mail and course websites weekly during the term and to note any changes. Your McMaster e-mail is the one with the address ending in @mcmaster.ca. This is a separate e-mail address from your Avenue address.