

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

May 25, 2024

This course outline contains important information that will affect 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, we discuss a systematic process for the development and documentation of requirements, system design, module design, implementation, testing, and inspection.

3 Introduction

Scientific computation (Research Software) 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 developers place little emphasis 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 developers sell scientific software 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) we identify a physical problem of engineering or scientific importance; ii) we derive a system of governing equations and the associated boundary conditions; iii) we develop numerical algorithms; iv) we implement the numerical algorithms on a computer; and, v) we verify and validate the model and the computed results, 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 we clearly document requirements (including assumptions) and design decisions.

Avenue, GitLab, GitHub and Teams

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, usernames 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 (CAS GitLab) repository. You can access this repository at:

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

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 at <https://github.com/>. Each student's repo will start from the given [template repo](#). Students will be expected to use GitHub to provide comments on the work of other students in the class.

We will use Teams for discussion. Our communication policy is that questions that are relevant to other students in the class should be posted on Teams. That way everyone can benefit from the discussion. Questions that are relevant to only one individual, should be sent to the instructor via e-mail. Please send only normal McMaster e-mail; do not send mail via Avenue or direct messages using Teams.

Classes

Classes will be held in person in the graduate classroom (ITB/222).

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. Typically, students will use one of two methodologies:

1. "Traditional/Manual Approach"
 - (a) Software Requirements Specification (SRS)
 - (b) Module Guide (MG)

- (c) Module Interface Specification (MIS)
 - (d) Implementation (any appropriate programming language)
 - (e) Verification and Validation (VnV) Plan
 - (f) Verification and Validation (VnV) Report
2. “Drasil (artifact+code generation)/Automated Approach”
- (a) Software Requirements Specification (SRS) draft, as for manual
 - (b) SRS generation in Drasil
 - (c) Drasil design choices and explanation
 - (d) Code generation (Drasil supported programming languages)
 - (e) Verification and Validation (VnV) Plan (manually produced)
 - (f) Verification and Validation (VnV) Report (manually produced)

If a student opts for the Drasil approach, they can later switch back to the Traditional approach, without any grade related penalty.

The Drasil approach uses the [Drasil framework](#) to generate the SRS and the code. Since Drasil is incomplete, the V&V documents will still be generated manually. Further details on Drasil will be covered during the class lectures.

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 a greater emphasis on domain specific code to automatically build the deliverables, or with extending Drasil’s capabilities. Projects that add something substantial to the base requirements are eligible for a bonus grade of up to 50%. For instance, adding support for a new external library to Drasil would be worth additional marks. The specific bonus should be discussed with the instructor in advance.

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, Drasil, Implementation, VnV Plan and VnV Report. The class discussion during each presentation will assist in improving the quality of the written deliverable. Given the term-long goal of improving each project, students should be prepared for criticism during their presentations. This criticism is part of formative learning; it is part of the learning process. It will not effect the presentation grades. Please let the instructor know if you have any concerns with the feedback you receive.

Each student following the traditional approach will be expected to virtually hand in the following written documents: SRS, MG, MIS, VnV Plan, VnV Report, code and Final Documentation. The deliverables for the Drasil approach are the same, except we replace the MG and MIS with the Drasil code to generate the documentation and code.

6 Grading

1. (Traditional and Drasil) Presentations and class discussion 5%
2. (Traditional and Drasil) “Domain Expert” and secondary reviewer roles 10%
3. (Traditional and Drasil) Problem Statement, Risk, Proof of Concept (POC) Plan 0%
4. (Traditional and Drasil) System Requirements Specification (SRS) Document 15%
5. (Traditional and Drasil) System Verification and Validation (VnV-Syst) Plan 15%
6. (Traditional) Module Guide and Module Interface Specification (MG and MIS) 15%
7. (Drasil) Drasil code and Code Explanation Document 15%
8. (Traditional and Drasil) Final Documentation 40%
 - (a) Problem Statement (Revised)
 - (b) SRS (Revised)
 - (c) System VnV Plan (Revised)
 - (d) (Traditional) MG (Revised)
 - (e) (Traditional) MIS (Revised)
 - (f) (Drasil) Drasil code and explanation (Revised)
 - (g) Unit VnV Plan
 - (h) Code
 - (i) System VnV Report
 - (j) Unit VnV Report
 - (k) Reflection Report

6.1 Peer Review

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 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.

6.2 Challenge Level and Extras

Due to differences in interests and background, students in this course will select topics with varying levels of complexity in both the required domain knowledge and in the implementation. Less advanced projects are an option, but to get full grades for a lower challenge project, students should add extra work to the scope of their project. Examples of extra work include usability testing, rigorous code walkthroughs, user manual documentation, formal proof, GenderMag personals, design thinking, etc.

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.

Conduct Expectations

As a McMaster graduate student, you have the right to experience, and the responsibility to demonstrate, respectful and dignified interactions within all of our living, learning and working communities. These expectations are described in the Code of Student Rights & Responsibilities (the “Code”). All students share the responsibility of maintaining a positive environment for the academic and personal growth of all McMaster community members, whether in person or online.

It is essential that students be mindful of their interactions online, as the Code remains in effect in virtual learning environments. The Code applies to any interactions that adversely affect, disrupt, or interfere with reasonable participation in University activities. Student disruptions or behaviours that interfere with university functions on online platforms (e.g. use of Avenue 2 Learn, WebEx or Zoom for delivery), will be taken very seriously and will be investigated. Outcomes may include restriction or removal of the involved students’ access to these platforms.

Academic Accommodation of Students with Disabilities

Students with disabilities who require academic accommodation must contact Student Accessibility Services (SAS) at 905-525-9140 ext. 28652 or sas@mcmaster.ca to make arrangements with a Program Coordinator. For further information, consult McMaster University’s Academic Accommodation of Students with Disabilities policy.

Academic Accommodation for Religious, Indigenous or Spiritual Observations (RISO)

Students requiring academic accommodation based on religious, indigenous or spiritual observances should follow the procedures set out in the RISO policy. Students should submit their request to their Faculty Office normally within 10 working days of the beginning of term in which they anticipate a need for accommodation or to the Registrar’s Office prior to their examinations. Students should also contact their instructors as soon as possible to make alternative arrangements for classes, assignments, and tests.

Copyright and Recording

Students are advised that lectures, demonstrations, performances, and any other course material provided by an instructor include copyright protected works. The Copyright Act and copyright law protect every original literary, dramatic, musical and artistic work, including lectures by University instructors

The recording of lectures, tutorials, or other methods of instruction may occur during a course. Recording may be done by either the instructor for the purpose of authorized

distribution, or by a student for the purpose of personal study. Students should be aware that their voice and/or image may be recorded by others during the class. Please speak with the instructor if this is a concern for you.

Extreme Circumstances

The University reserves the right to change the dates and deadlines for any or all courses in extreme circumstances (e.g., severe weather, labour disruptions, etc.). Changes will be communicated through regular McMaster communication channels, such as McMaster Daily News, A2L and/or McMaster email.