

CAS 741, CES 741 (Development of Scientific Computing Software)

Fall 2019

01 Introduction

Dr. Spencer Smith

Faculty of Engineering, McMaster University

September 12, 2019



Introduction to CAS 741 (CES 741)

- Administrative details
- Brief overview of course
- Introductions
- Course outline
- Requirements

Administrative Details

- New Grad Stdnt Orientation: Mon, Sept 9, 10 am – 1 pm
- Lectures: Mon and Thurs, 10:30 am – 12:00 noon
- Avenue for grade tracking
 - ▶ <http://avenue.mcmaster.ca/>
 - ▶ Consider putting a picture up on Avenue
- We'll also use git on GitLab for the course material
 - ▶ <https://gitlab.cas.mcmaster.ca/>
 - ▶ Create your account by logging in, option to set CAS password to MacID password
 - ▶ Course material and issue tracking at <https://gitlab.cas.mcmaster.ca/smiths/cas741>
- Your projects will be hosted on GitHub
 - ▶ <https://github.com/>
 - ▶ Create an account, if you do not already have one
 - ▶ Give the instructor (me) master access to your repo

Overview of the Course

- Application of software engineering methodologies to improve the quality of scientific computing software
- What is the definition of scientific computing?
- What are some examples of scientific computing and scientific computing software?
- What is the definition of software engineering?
- What are some techniques, tools and principles for software engineering?

Scientific Computing (SC)

- 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.
- Examples
 - ▶ Temperature of fuel-pin in nuclear reactor
 - ▶ Flow of pollutant in groundwater
 - ▶ Displacement of a structure
 - ▶ Thickness of cast film
 - ▶ Temperature of water in a solar water heating tank over time
 - ▶ Ordinary Differential Equation solver
 - ▶ Root finding solver etc.
- Includes analysis, design and “exploration”

Software Engineering (SE)

- SE is an area of engineering that deals with the development of software systems that
 - ▶ Are large or complex
 - ▶ Exist in multiple versions
 - ▶ Exist for large period of time
 - ▶ Are continuously being modified
 - ▶ Are built by teams
- SE is “application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software” (IEEE 1990)
- D. Parnas (1978) defines SE as “multi-person construction of multi-version software”
- Like other areas of engineering, SE relies heavily on mathematical techniques (logic and discrete math)
- SE might be applied to SC for software certification

SE Tools, Techniques and Principles

- Tools
 - ▶ Programming languages
 - ▶ Version control software (git, svn, etc)
 - ▶ Debugger
 - ▶ Profiler
 - ▶ ...
- Techniques
 - ▶ Documentation
 - ▶ Testing
 - ▶ Program families
 - ▶ Code generation
 - ▶ ...
- Principles
 - ▶ Information hiding
 - ▶ Least privilege
 - ▶ ...

Instructor

- Instructor
 - ▶ Dr. Spencer Smith (smiths@mcmaster.ca)
 - ▶ ITB/167
 - ▶ Drop in or make an appointment
 - ▶ Feel free to make use of the instructor's [schedule](#)

Introduction: Dr. Spencer Smith

- Associate Professor, Department of Computing and Software.
- B.Eng.C.S, Civil Engineering Department, McMaster University.
M.Eng., Ph.D., Civil Engineering Department, McMaster University.
- P.Eng. (Licensed Professional Engineer in Ontario).
- **Teaching:** Software design, scientific computing, introduction to computing, communication skills, software project management.
- **Research:** Application of software engineering methodologies to improve the quality of scientific computing software.

Introductions

- Your name
- Degree program, supervisor's name
- Academic background
- Experience with:
 - ▶ Science (such as physics)
 - ▶ Scientific computing
 - ▶ Continuous math
 - ▶ Discrete math
 - ▶ Software engineering
 - ▶ Software development technology
 - ▶ Git
 - ▶ GitHub or GitLab
 - ▶ LaTeX
 - ▶ Make etc.
- What do you hope to get out of this course?

Course Introduction

- Calendar description
 - ▶ Principles of software development for reliable and sustainable scientific and engineering software
 - ▶ Systematic process for development and documentation of
 - ▶ Requirements
 - ▶ System architecture
 - ▶ Detailed design
 - ▶ Implementation
 - ▶ Verification and Validation Plan
 - ▶ Verification and Validation Report

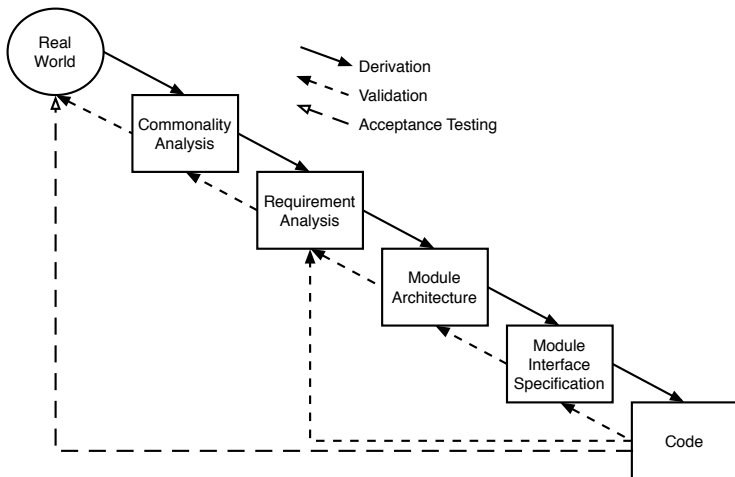
Course Project

- Select a candidate SC problem
 - ▶ Requires approval from instructor
 - ▶ Will accommodate your interests as much as feasible
 - ▶ Select a project related to your research
 - ▶ Scope needs to be feasible within one term
- Milestones
 1. Software Requirements Specification (SRS)
 2. Module Guide (MG)
 3. Module Interface Specification (MIS)
 4. Implementation (and appropriate programming language)
 5. VnV Plan
 6. VnV Report
- Deliverables can potentially be modified to provide project flexibility

Project Selection: Desired Qualities

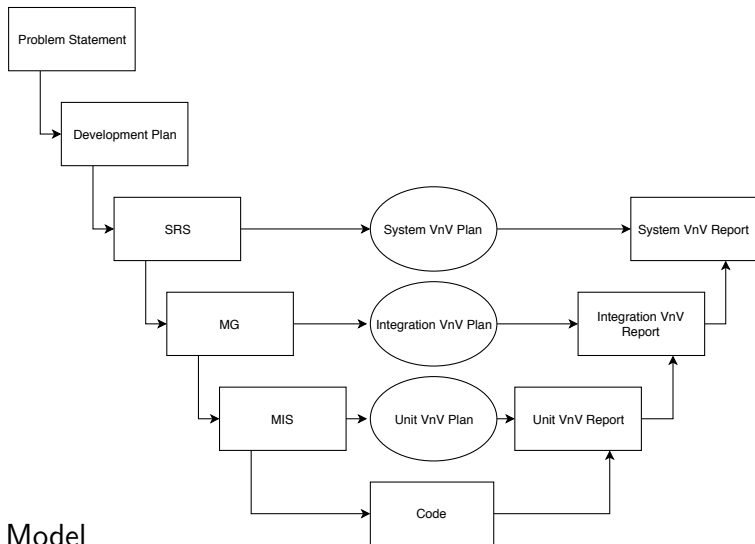
- Related to scientific computing
- Simple, but not trivial
- If feasible, select a project related to your research
- Possibly re-implement existing software
- Each student project needs to be unique
- Possibly a specific physical problem
- Possibly a (family of) general purpose tool(s)
- Some examples follow, the links are just places to get started

“Faked” Rational Design Process



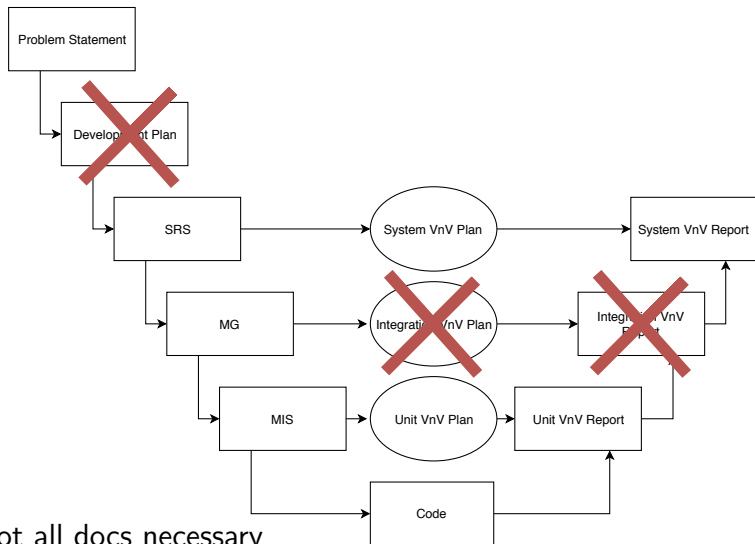
See Parnas and Clements 1986 about “Faking It”

Our “Faked” Process



V Model

Our Deliverables



Not all docs necessary

Course Structure

- Student and instructor presentations
- Classroom discussions
- Will present a subset of your documentation for in-class feedback
- Structure from our documentation
- Use GitHub issue tracker for feedback from other students

Grade Assessment

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 (including revised versions of previous documents, plus the source code, unit testing plan, reflection report and testing reports (System and Unit)) 40%
8. Drasil simple physics example, pull request accepted 5% (Bonus)

Policy Statements

- Ideas to improve the course are welcomed
- Missed/late work please communicate in advance, or a penalty of 20 % per working day
- If there is a problem with discrimination please contact the Department Chair, or other appropriate body

Academic Dishonesty

- Academic dishonesty consists of misrepresentation by deception or by other fraudulent means
- Can result in serious consequences, e.g. the grade of zero on an assignment, loss of credit with a notation on the transcript, and/or suspension or expulsion from the university.
- It is your responsibility to understand what constitutes academic dishonesty
- Three examples of academic dishonesty
 - ▶ Plagiarism
 - ▶ Improper collaboration
 - ▶ Copying or using unauthorized aids in tests and examinations
- Academic dishonesty will not be tolerated!

Assigned Reading

As often as possible, hyperlinks are included for references in the lecture slides

- [W. Spencer Smith](#). A rational document driven design process for scientific computing software.
In Jeffrey C. Carver, Neil Chue Hong, and George Thiruvathukal, editors, *Software Engineering for Science*, chapter Section I – Examples of the Application of Traditional Software Engineering Practices to Science, pages 33–63. Taylor & Francis, 2016
- [W. Spencer Smith, Lei Lai, and Ridha Khedri](#).
Requirements analysis for engineering computation: A systematic approach for improving software reliability.
Reliable Computing, Special Issue on Reliable Engineering Computation, 13(1):83–107, February 2007

Assigned Reading

- David L. Parnas and P.C. Clements. A rational design process: How and why to fake it.
IEEE Transactions on Software Engineering, 12(2): 251–257, February 1986
- Solar Water Heating System Example
- Solar Water Heating System Example SRS (Generated by Drasil)
- Projectile Example SRS (Generated by Drasil)
- Recommended reading order for SRS documents
 - ▶ Goal Statement
 - ▶ Instance Models
 - ▶ Requirements
 - ▶ Introduction
 - ▶ Specific System Description