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

## Fall 2017

# 01 Introduction

Dr. Spencer Smith

Faculty of Engineering, McMaster University

September 4, 2017

# Introduction to CAS 741 (CES 741)

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

# Administrative Details

- Lecture times
  - Tuesdays, 9:00 am to 10:30 am
  - Fridays, 9:00 am to 10:30 am
- This course uses Avenue
  - 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
  - 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
  - Access to your repo to instructor, all students in the class, your supervisor?, other interested parties?

# 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 engineeing?
- 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 systems 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
  - etc.

# Software Engineering (SE)

- 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
- Software engineering is "application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software" (IEEE 1990)
- D. Parnas (1978) defines software engineering as "multi-person construction of multi-version software"
- Like other areas of engineering, software engineering relies heavily on mathematical techniques, especially logic and discrete mathematics
- SE might be applied to SC for software certification

# Instructor

- Instructor
  - Dr. Spencer Smith (smiths@mcmaster.ca)
  - ITB/167
  - Drop in or make an appointment

# Introduction of Instructor: 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
- Academic background
- Experience with:
  - 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?

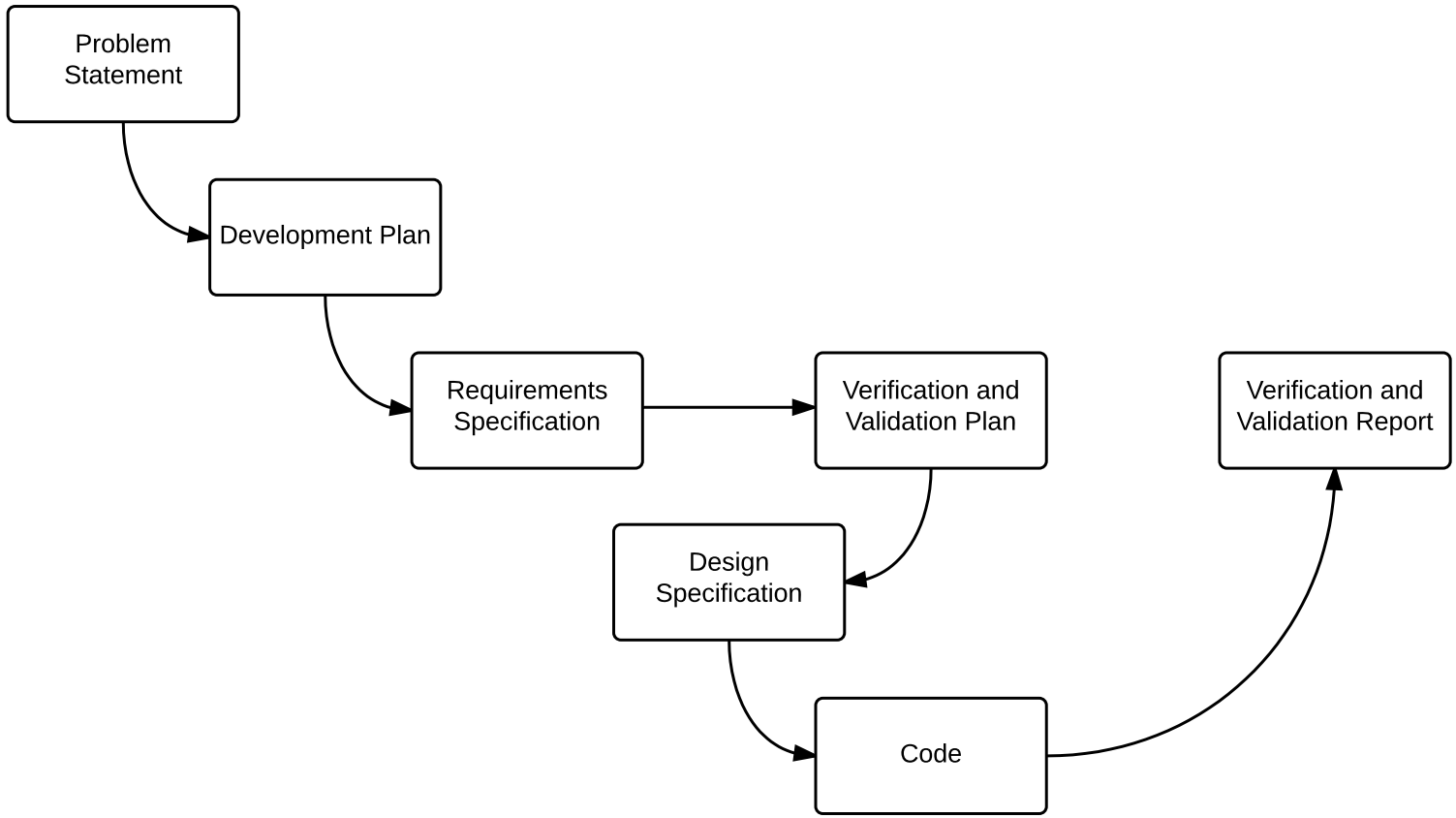
# Course Introduction

- Calendar description
  - Principles of software development for reliable scientific and engineering software
  - Systematic process for development and documentation of
    - Requirements
    - System architecture
    - Detailed design
    - Implementation
    - Testing and inspection

# 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. Testing
- Deliverables can potentially be modified to provide project flexibility

# "Faked" Rational Design Process




See Parnas and Clements 1986 about "Faking It"

# 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 10%
   - Assigned presentations
   - In-class discussion
2. Quality of GitHub issues provided to classmates 5%
3. System Requirements Specification (SRS) 20%
4. Module Guide (MG) 10%
5. Module Interface Specification (MIS) 20%
6. Final Documentation (including revised versions of previous documents, plus the source code and a testing report) 35%

# Policy Statements

- Ideas to improve the course are welcomed
- Missed/late work use MSAF, 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!