

# SE 2AA4, CS 2ME3 (Introduction to Software Development)

Winter 2018

## 01 Introduction to the Course

Dr. Spencer Smith

Faculty of Engineering, McMaster University

January 9, 2018



# 01 Introduction to the Course

- Administrative details
- Technology used
- Course outline
  - ▶ Introduction
  - ▶ Learning objectives
  - ▶ Outline of topics
  - ▶ Grade assessment
  - ▶ Policy statements

# Benches and Glassboards



# Administrative Details

- Backpack to Briefcase
- Big Ideas Contest
- On-line Course Permission Waiver Form
- Midterm: Wed, February 28, 7:00 – 8:30 pm in T13 106, T13 123, and UH 213
- Assignment 1
  - ▶ Part 1 due by January 22, 11:59 pm
  - ▶ Partner files by January 28
  - ▶ Part 2 (lab report) due by January 31, 11:59 pm
  - ▶ Will still be minor revisions, communicated in class
  - ▶ See [Previous Assignment](#)
- Tutorials start next week - bring your laptop

# Administrative Details Continued

- This course uses Avenue
  - ▶ <http://avenue.mcmaster.ca/>
  - ▶ Please put a picture up on Avenue!
- We'll also use git on GitLab
  - ▶ <https://gitlab.cas.mcmaster.ca/>
  - ▶ Create your account by logging in
  - ▶ Can set CAS password to MacID password using <https://www.cas.mcmaster.ca/macid>
  - ▶ Course material and issue tracking at [https://gitlab.cas.mcmaster.ca/smiths/se2aa4\\_cs2me3](https://gitlab.cas.mcmaster.ca/smiths/se2aa4_cs2me3)
    - ▶ Repo is always a work in progress
    - ▶ Make sure you are looking at the most recent commit!
    - ▶ Feel free to assign me issues
- Please register on GitLab before your first tutorial

# Technology Used in 2AA4/2ME3

- Python
- C++
- Git
- GitLab
- LaTeX
- Doxygen
- Make
- `ssh [CASid]@mills.cas.mcmaster.ca`
- Command line or terminal
- [CAS Support Wiki](#)

# Very Short Git Primer

```
git clone https://gitlab.cas.mcmaster.ca/smiths/  
se2aa4_cs2me3.git
```

```
git clone https://gitlab.cas.mcmaster.ca/  
se2aa4_cs2me3_assignments_2018/[macid].git
```

```
git pull
```

```
git add
```

```
git commit
```

```
git push
```

# Instructors

- Instructor
  - ▶ Dr. Spencer Smith ([smiths@mcmaster.ca](mailto:smiths@mcmaster.ca))
  - ▶ ITB/167
  - ▶ Drop in (see [schedule](#)) or make an appointment
- TAs
  - ▶ Steven Palmer ([palmes4@mcmaster.ca](mailto:palmes4@mcmaster.ca))
  - ▶ Henry Madej ([madejh@mcmaster.ca](mailto:madejh@mcmaster.ca))
  - ▶ Sepehr Bayat ([bayats1@mcmaster.ca](mailto:bayats1@mcmaster.ca))
  - ▶ Zichen Jiang ([jiangz26@mcmaster.ca](mailto:jiangz26@mcmaster.ca))
  - ▶ Justin Staples ([staplejw@mcmaster.ca](mailto:staplejw@mcmaster.ca))
  - ▶ Teaching assistants will
    - ▶ Give tutorials
    - ▶ Mark assignments
    - ▶ Provide design/programming/technical assistance
    - ▶ Answer questions on the course material

# 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.

# Introduction

- Calendar description
  - ▶ Development of small software units
  - ▶ Precise specification using logic and discrete math
  - ▶ Design methods and design patterns
  - ▶ Implementation and testing
- Mission
  - ▶ Introduction to profession of software engineering
  - ▶ Strategies for large applications with multiple developers
  - ▶ Python and C++

# Learning Objectives

<https://gitlab.cas.mcmaster.ca/.../LearningOutcomes>

# Resources

- Ghezzi et al. (2003) (required)
- [Hoffman and Strooper \(1995\)](#) (other)
- VanVliet (2000) (other)

# Outline of Topics

1. Introduction to Course
2. Software Engineering as an Engineering Discipline [Chapter 1]
3. Software Qualities [Chapter 2]
4. Software Engineering Principles [Chapter 3]
5. Software Design [Chapter 4]
6. Modularization [Chapter 4]
7. Specification [Chapter 5]
8. Verification [Chapter 6]
9. The Software Development Process [Chapter 7]
10. Design Patterns

# Grade Assessment

1. Assignments 30%
2. Midterm 25%
3. Final Exam 45%

# Assignments

- Four equally weighted assignments
- Assignments must be your own work
- **Do not allow other students to copy your work**
- Explicitly cite all sources, including:
  - ▶ All on-line resources
  - ▶ Your fellow students
- Keep all of your working notes and files used to prepare your solutions
- If there is a problem with a grade
  - ▶ Report it first to the TA
  - ▶ Report it within two weeks of receiving your grade
- The assignment grade will only be counted if the weighted average of the midterm and final is greater than 50 %

# Examinations

- Midterm
  - ▶ 90 minutes
  - ▶ Multiple choice
  - ▶ Bring your student card
  - ▶ Bring pencil
- Final examination
  - ▶ 2.5 hours
  - ▶ Multiple choice and written answers
  - ▶ Scheduled by registrar
  - ▶ Will cover entire course

# Policy Statements

- No calculators
- Ideas to improve the course are welcomed
- Missed/late work use MSAF
  - ▶ 5 day extension for assignments
  - ▶ Midterm rolled into final
- If there is a problem with discrimination please contact the Department Chair, or other appropriate body

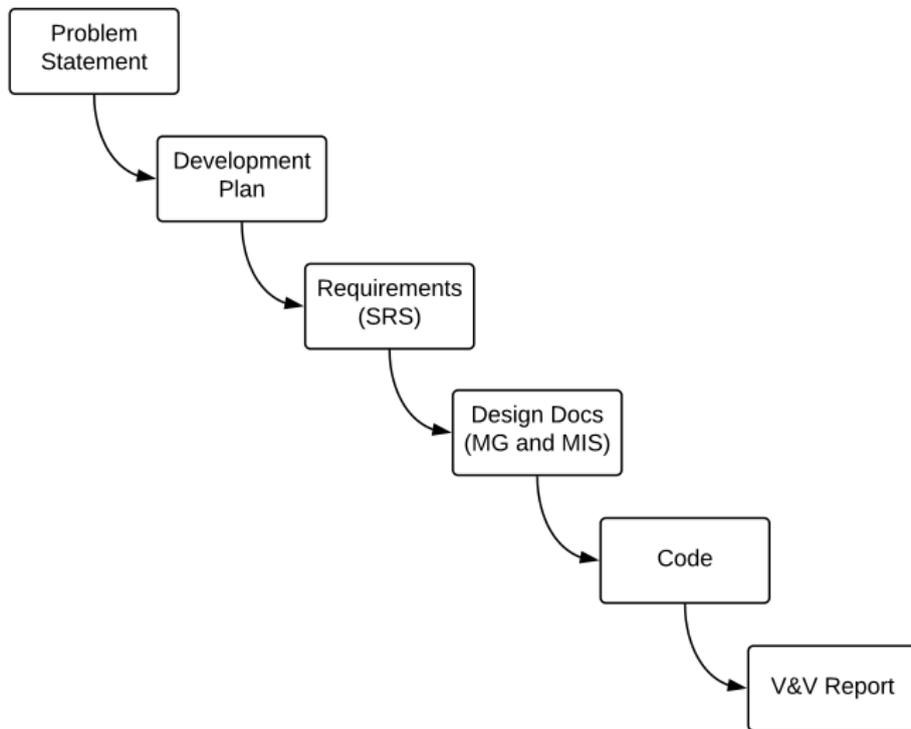
# Academic Dishonesty (AD)

- Knowingly act or fail to act in a way that results, or could result in, unearned academic credit or advantage
- Serious consequences
- Your responsibility to understand what constitutes AD
- Three examples
  - ▶ Plagiarism
  - ▶ Improper collaboration
  - ▶ Copying or using unauthorized aids in tests
- Cite all sources! including on-line and fellow students
- If in doubt – cite!
- Academic dishonesty will not be tolerated!
- 27 cases investigated last year

# Course Evaluations

<b>Class Participation</b>	<b>Bonus</b>
80–84%	0.75
85–89%	1.00
90–94%	1.25
95–100%	1.50

# “Faked” Rational Design Process



See [Parnas and Clements 1986](#) about “Faking It”

# Software Engineering versus Computer Science

What is the difference between Software Engineering and Computer Science?

# SE versus CS

- CS is the applied science for SE
- CS for theory, models
- SE applies theory and models
- SE associated with large, complex software
- SE associated with design and documentation
- SE associated with project management
- SE associated with real time embedded systems
- SE has an explicit ethical component
- Really a continuum

# What is Software Engineering?

- 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

# Interesting Article

Programmers Should Not Call Themselves Engineers