Course Overview

CSCI 5828: Foundations of Software Engineering
Lecture 01 — 01/17/2012

Goals

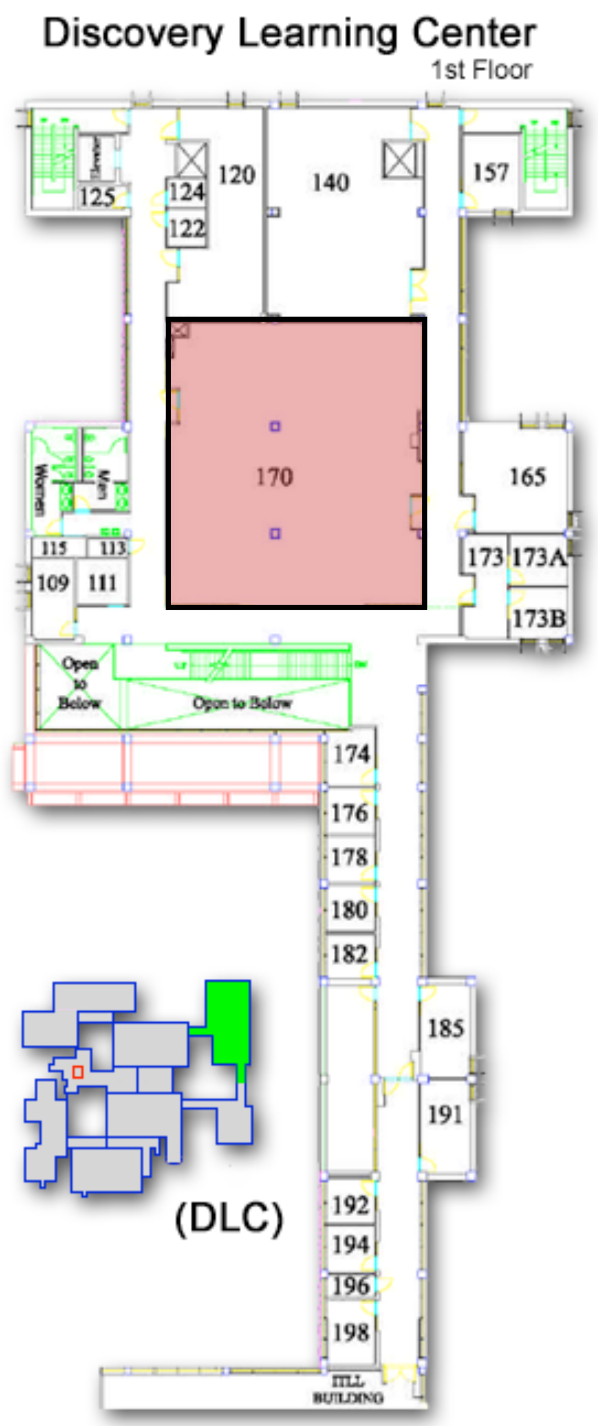
- Present a fundamental introduction to the field of software engineering
    - Present brief history and foundational theory of software engineering
    - Survey software engineering concepts, terminology, and techniques
- Take an in-depth look at three important software engineering concepts
    - software development life cycles, with an emphasis on agile methods
    - designing and implementing concurrent software systems
    - software testing and how to integrate it throughout the life cycle

About Me

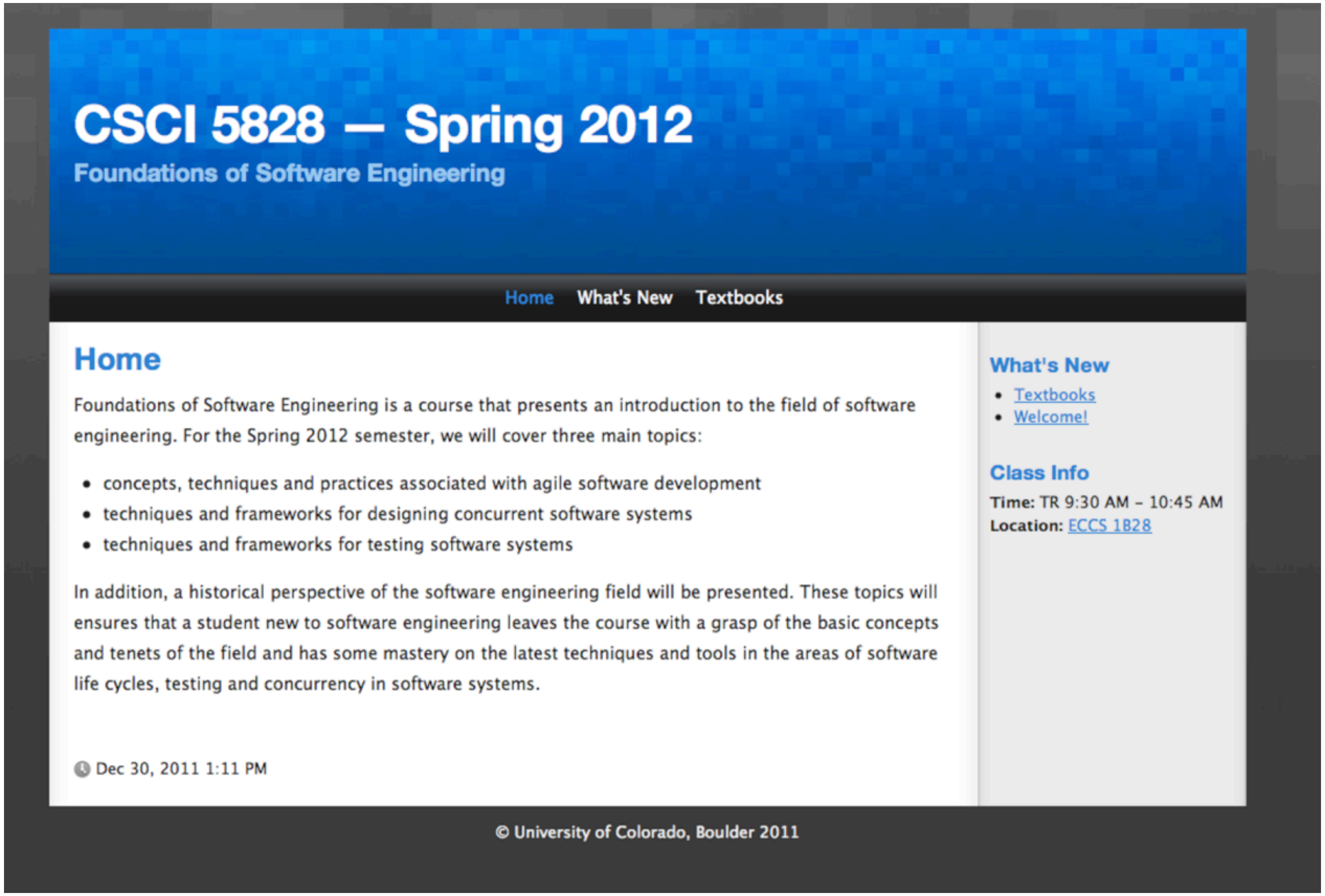
- Associate Professor
  - Ph.D. at UC Irvine
  - 13.5 Years at CU;
    - Start of my 28th Semester!
- 7th time teaching this class
- Research Interests
  - Software & Web Engineering
  - Software Architecture
  - Hypermedia

Office Hours

- Thursdays, 1 PM to 3 PM, or by appointment
    - DLC 170 (shown in red on right)
- Please send me e-mail to let me know you plan to stop by



Class Website


CSCI 5828 — Spring 2012

Foundations of Software Engineering

Home What's New Textbooks

Home

Foundations of Software Engineering is a course that presents an introduction to the field of software engineering. For the Spring 2012 semester, we will cover three main topics:

- concepts, techniques and practices associated with agile software development
- techniques and frameworks for designing concurrent software systems
- techniques and frameworks for testing software systems

In addition, a historical perspective of the software engineering field will be presented. These topics will ensures that a student new to software engineering leaves the course with a grasp of the basic concepts and tenets of the field and has some mastery on the latest techniques and tools in the areas of software life cycles, testing and concurrency in software systems.

Dec 30, 2011 1:11 PM

What's New

- Textbooks
- Welcome!

Class Info

Time: TR 9:30 AM – 10:45 AM

Location: ECCS 1B28

© University of Colorado, Boulder 2011

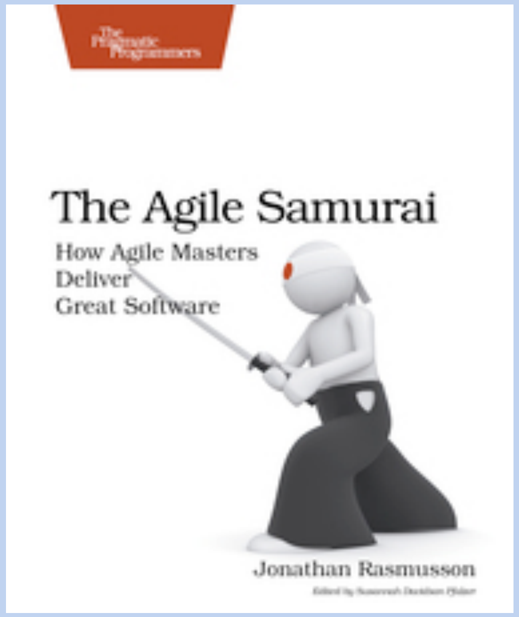
<http://www.cs.colorado.edu/~kena/classes/5828/s12/>

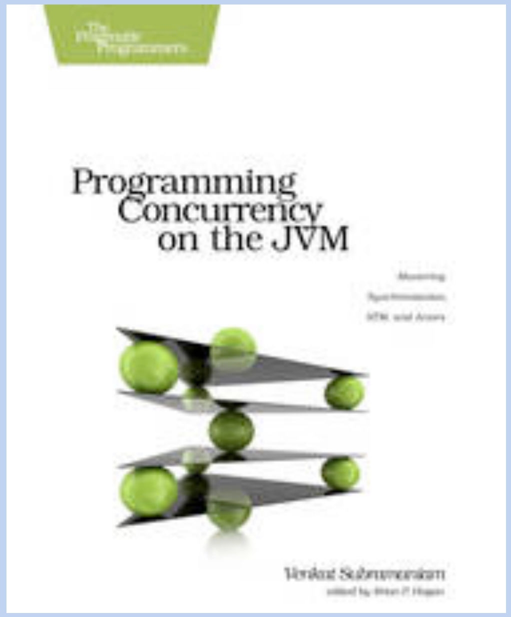
Check the website every day! (I’m serious)

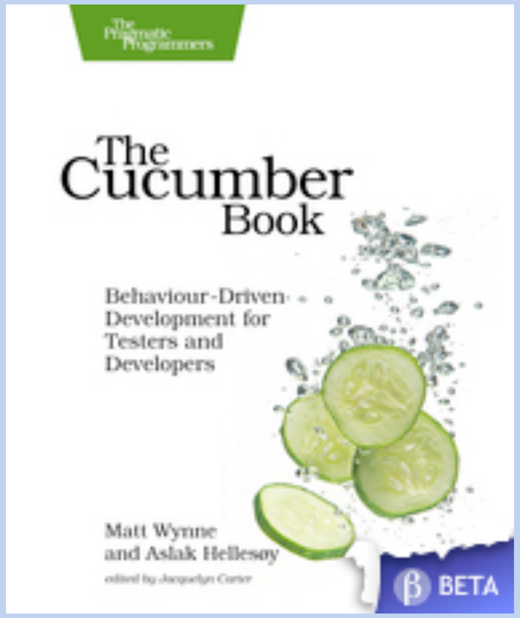
- To make it easy for you to track updates
    - Go to the “What’s New” page and
    - Subscribe to the RSS feed
- Feed readers are available for all platforms
    - NetNewsWire, FeedDemon, Google Reader, etc.
- The website is your source for
    - the class schedule, homework assignments, announcements, etc.

Textbooks

Available at http://pragprog.com/ or our fine bookstore


The Pragmatic Programmers
The Agile Samurai
How Agile Masters Deliver Great Software
Jonathan Rasmusson




The Pragmatic Programmers
The Cucumber Book
Behaviour-Driven Development for Testers and Developers
Matt Wynne and Aslak Hellesøy
BETA

Required

- The Agile Samurai by Jonathan Rasmusson
- Programming Concurrency on the JVM by Venkat Subramaniam

Optional

- The Cucumber Book by Matt Wynne and Aslak Hellesøy

Tentative Course Structure

	Dates	Tuesday Topic	Thursday Topic	Due Thursday
Week 1	Jan 17th, 19th	Overview	Fred Brooks	Read No Silver Bullet
Week 2	Jan 24th, 26th	Intro Life Cycles	Concurrency 1	Homework 1
Week 3	Jan 31st, Feb. 2nd	Intro Testing	Agile 1, 2	Quiz 1
Week 4	Feb. 7th, 9th	Concurrency 2, 3	Testing 1, 2, 3	Homework 2
Week 5	Feb. 14th, 16th	Agile 3, 4, 5	Concurrency 4	Quiz 2
Week 6	Feb. 21st, 23rd	Testing 4, 5, 6	Agile 6, 7, 8	Homework 3
Week 7	Feb. 28th, Mar. 1st	Concurrency 5	Midterm Prep	Quiz 3
Week 8	Mar. 6th, 8th	Midterm	Testing 7, 8	
Week 9	Mar. 13th, 15th	Agile 9	Concurrency 6, 7	Quiz 4
Week 10	Mar. 20th, 22nd	Testing 9, 10	Agile 10, 11	Homework 4; Presentation
Spring Break	Mar. 27th, 29th	Sleeping	Sunbathing	
Week 11	Apr. 3rd, 5th	Concurrency 8, 9	Testing 11, 12	Quiz 5
Week 12	Apr. 10th, 12th	Agile 12, 13	Concurrency 10	Homework 5
Week 13	Apr. 17th, 19th	Testing 13, 14	Agile 14, 15	Quiz 6
Week 14	Apr. 24th, 26th	Concurrency: GCD	Testing, 15, 16	Homework 5
Week 15	May 1st, 3rd	TBA	TBA	Final Project

Emphasis on Tentative

- The schedule on the previous slide may change
  - For instance, I do not have a time on the schedule to review the results of the midterm
    - (That's because I can't predict how fast I'll get them graded)
- However, you can trust that
  - There will be homeworks, quizzes, a midterm, a presentation, and a project
  - The midterm will be held on Tuesday, March 6th
    - CAETE students will need to work with CAETE to identify a person to proctor their midterm exam; You will have from March 6th to March 13th (one week) to take your exam and have it sent to me by your proctor

Course Evaluation

- Your grade will be determined by your work on
  - Class Participation and Attendance (5%)
  - Quizzes (15%)
  - Homeworks (20%)
  - Midterm (20%)
  - Presentation (20%)
  - Project (20%)
- Quizzes will be taken on the moodle; the presentation and project will be submitted on the moodle as well
- Homeworks are being treated differently this semester (discussed soon)

Honor Code

- You are allowed to work together in teams of up to 2 people on
    - the homeworks
    - the presentation
    - the project
- The quizzes and the midterm are individual work

- The Student Honor Code applies to classes in all CU schools and colleges. You can learn about the honor code at:
    - <http://www.colorado.edu/academics/honorcode/>.

Late Policy

- Assignments submitted late incur a 15% penalty
  - You may submit a homework assignment and the presentation up to one week late
    - after that the submission will not be graded and you'll receive 0 points for it
  - The quizzes, the midterm, and the project may not be submitted late
    - If you discover that you cannot attend the midterm on March 6th, you need to get in touch with me ASAP **before** the midterm to make other arrangements
      - trying to make arrangements **after** the midterm will be very difficult

Homeworks

Tentative Plan; I’m open to feedback

- I’m going to take a different approach with homeworks this semester
  - I’m going to have **you** participate in grading them during class
  - On the day a homework assignment is due, you will bring a hard copy of your assignment to class; the hardcopies will be distributed amongst your peers
    - if you work in a team, bring only one hard copy for the team
    - if you work in a team, you will grade as a team
  - I will provide a grading rubric
  - I will review the answers to the assignment
  - You will assign a preliminary grade to the assignment
  - The graded assignments will then be handed in and the grader will review them and assign final grades

Discussion on Plans for Homeworks

- This process is new for me. It will require iteration to get it right.
  - Please be flexible and open to making this work
- My goal in doing this is to increase your own knowledge of the course material
- CAETE students will do this at home for their peers
  - You'll submit assignments to the moodle and I'll redistribute them
  - You'll grade at home after you watch the lecture and then send the assignment with your preliminary grade to the grader
- The grader is involved in the process to guard against students who are either too harsh or too lenient in their grading

Syllabus Statements

- The University asks that various policies be presented to students at the start of each semester. These policies include
    - Disability Accommodations
    - Religious Observances
    - Classroom Behavior
    - Discrimination and Harassment
    - Honor Code
- See <http://www.cs.colorado.edu/~kena/classes/5828/s12/syllabus-statements.html> for more details

Programming Languages

- Code examples this semester will be drawn from a number of languages
    - Java
    - Objective-C
    - Ruby
    - Python
- In general, I'm agnostic on programming languages used for assignments
    - However, some of your homework assignments will require Java as we'll be studying its concurrency features fairly closely

What is Software Engineering

- **Software**
  - Computer programs and their related artifacts
    - e.g. requirements documents, design documents, test cases, UI guidelines, usability tests, …
- **Engineering**
  - The application of scientific principles in the context of practical constraints
- Consider: **Chemist versus Chemical Engineer**
  - Software engineers have a similar relationship with computer scientists
  - Software engineering has a similar relationship with computer science

Emphasizing the Point

- Consider this recent story on Slashdot:
  - IBM Shrinks Bit Size To 12 Atoms
- From the story:
  - “IBM researchers say they've been able to shrink the number of iron atoms it takes to store a bit of data from about one million to 12… Andreas Heinrich, who lead the IBM Research team on the project for five years, said the team used the tip of scanning tunneling microscope and unconventional antiferromagnetism to change the bits from zeros to ones… That **solved a theoretical problem** of how few atoms it could take to store a bit; **now comes the engineering challenge**: how to make a mass storage device perform the same feat as scanning tunneling microscope.

What is Software Engineering

- What is **Engineering**?
  - Engineering is a sequence of well-defined, precisely-stated, sound steps, which follow a method or apply a technique based on some combination of
    - theoretical results derived from a formal model
    - empirical adjustments for unmodeled phenomenon
    - rules of thumb based on experience
- This definition is **independent of purpose**
  - i.e. engineering can be applied to many disciplines

What is Software Engineering

- Software engineering is that form of engineering that applies…
  - a systematic, disciplined, quantifiable approach,
  - the principles of computer science, design, engineering, management, mathematics, psychology, sociology, and other disciplines…
- to creating, developing, operating, and maintaining cost-effective, reliably correct, high-quality solutions to software problems. (Daniel M. Berry)

- With respect to disciplined
  - Consider: Difference between professional musician and amateur musician

What is Software Engineering?

- Issues of Scale
    - Software engineers care about developing techniques that enable the construction of large scale software systems
- Issues of Communication
    - Consider the set of tools provided by sites like Rally, Fogbugz, or Assembla.com
- Issues of Regulation
    - Other engineering disciplines require certification; should SE?
- Issue of Design
    - dealing with integration of software/hardware/process

Types of Software Development

- Desktop Application Development
- Contract Software Development / Consulting
- Mobile Application Development
- Web Engineering (Development of Web Applications)
- Military Software Development
- Open Source Software Development
- Others??
  - These categories are not orthogonal!

Jobs related to Software Engineering

- Software Developer
- Software Engineer
- SQA (Software Quality Assurance) Engineer
- Usability Engineer
    - requires strong HCI/CSCW background
- Systems Analyst
    - professional requirements gather and/or designer
- DBA
- System administrator
- Software Architect
- Software Consultant
- Web Designer
- Build Manager / Configuration Management Engineer
- Systems Engineer
- Computer Graphics Animator

Core Principles (What I call “The Big Three”)

- **Specification**
  - Software engineers specify **everything**
    - requirements, design, code, test plans, development life cycles
    - What makes a good specification?
- **Translation**
  - The work of software engineering is one of **translation**, from one **specification** to another; from one level of **abstraction** to another; from one set of **structures** to another (e.g. problem/design decomposition)
- **Iteration**
  - The work of software engineering is done iteratively; step by step until we are “done”

These Core Principles are Everywhere

- You will find these principles in all things related to software engineering
  - its techniques & tools
  - its development life cycles
  - its practices
- And the most important part of software engineering?
  - The people who perform it
- Ultimately, software engineering comes down to the people involved
  - the customers, the developers, the designers, the testers, the marketers, etc.; You'll find the best development projects are **conversations**

Software Engineering: More than just Programming

- Sample Application with a performance problem
  - A program which loads a directory of images and animates them
  - It takes a while for the images to appear
- How would a software engineer go about discovering what part of the code is slowing things down?
- The fix will often not be easy
  - Case in point: to fix the problem, we needed to introduce concurrency
    - and that can introduce its own problems!

Software Engineering: More than just Programming

- Discussion
  - We could have used “print” statements to diagnose problem
    - but that introduces a bunch of code that we have to delete later
      - problem: we might make a mistake when adding those statements or taking them away ⇒ i.e. maintenance headaches
    - plus, we’re just duplicating (badly) the functionality that more powerful tools can provide
      - debuggers and profilers exist: software engineers use them
  - Solutions are not straightforward and will trigger re-designs
    - Adding a queue and chunking the work into tasks is not easy

Software Engineering is Hard

- No doubt about it: software engineering is hard
  - Projects are late, over budget, and deliver faulty systems
- See 1995 Standish Report for one summary of the problem
- Why?
  - For insight, we will take a look at an article by Fred Brooks called No Silver Bullet
  - **Please read it by Thursday's lecture**
    - Paper is available on IEEE Digital Library: No Silver Bullet

Questions?

Coming Up Next

- Lecture 2: No Silver Bullet
- Lecture 3: Introduction to Software Life Cycles
- Lecture 4: Introduction to Concurrent Software Systems
- Lecture 5: Introduction to Software Testing