Course Overview

CSCI 5828: Foundations of Software Engineering
Lecture 01 — 08/26/2014

All problems in computer science can be solved by another level of indirection.

David Wheeler

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 design

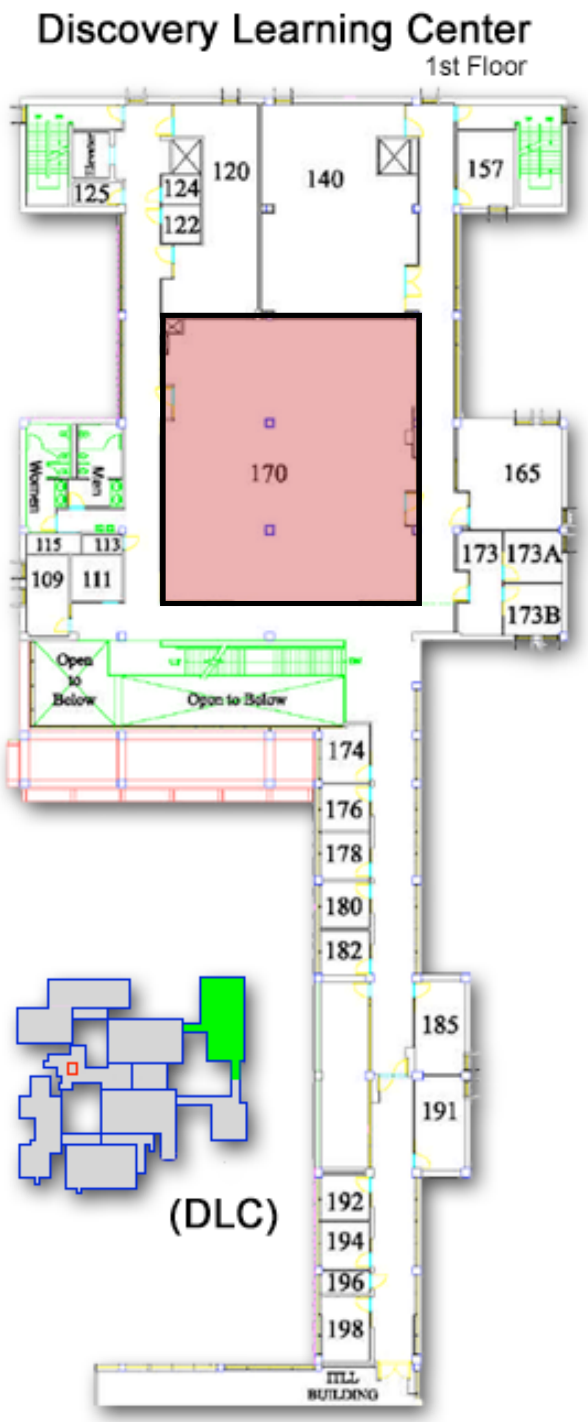
About Me

- Associate Professor
    - Ph.D. at UC Irvine
    - 16 Years at CU;
        - Start of my 33rd Semester!
- 8th time teaching this class
- Research Interests
    - Software & Web Engineering
    - Software Architecture
    - Crisis Informatics

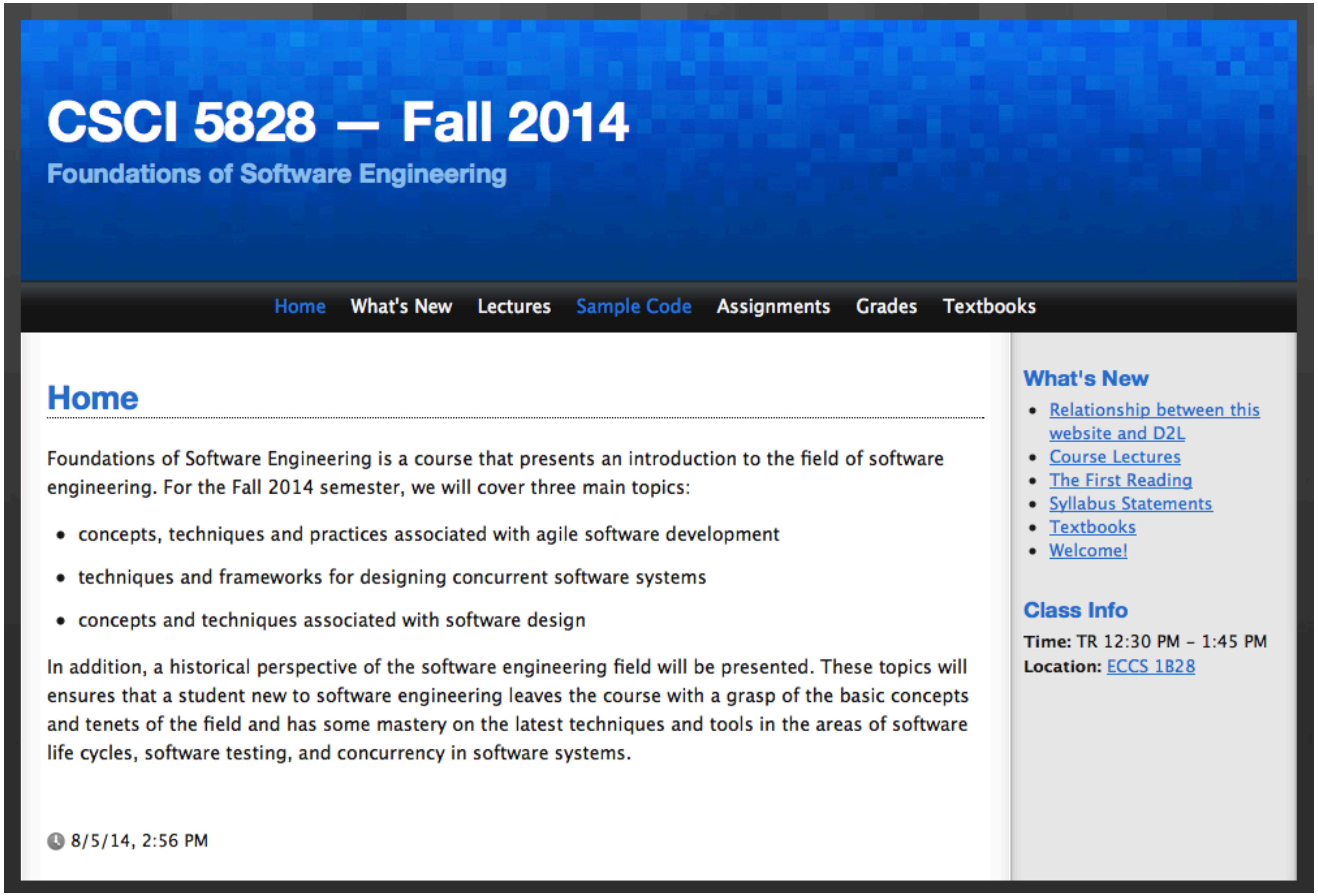
Office Hours

- Fridays, 2 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 — Fall 2014

Foundations of Software Engineering

Home

Foundations of Software Engineering is a course that presents an introduction to the field of software engineering. For the Fall 2014 semester, we will cover three main topics:

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

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, software testing, and concurrency in software systems.

8/5/14, 2:56 PM

What's New

- Relationship between this website and D2L
- Course Lectures
- The First Reading
- Syllabus Statements
- Textbooks
- Welcome!

Class Info

Time: TR 12:30 PM – 1:45 PM
Location: ECCS 1B28

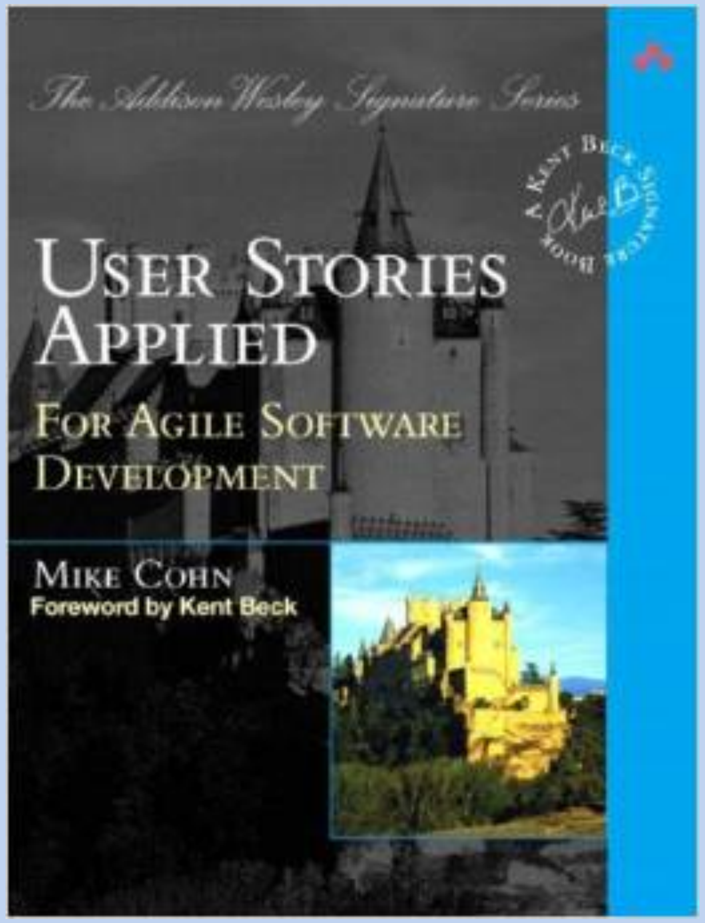
<http://www.cs.colorado.edu/~kena/classes/5828/f14/>

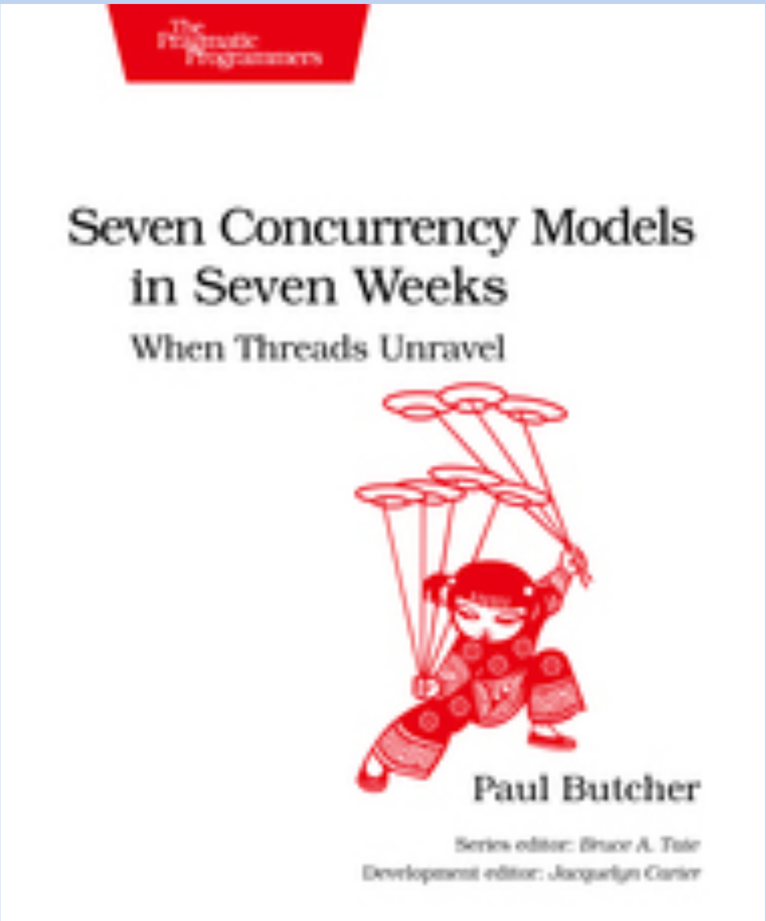
Check the website every day! (I'm serious)

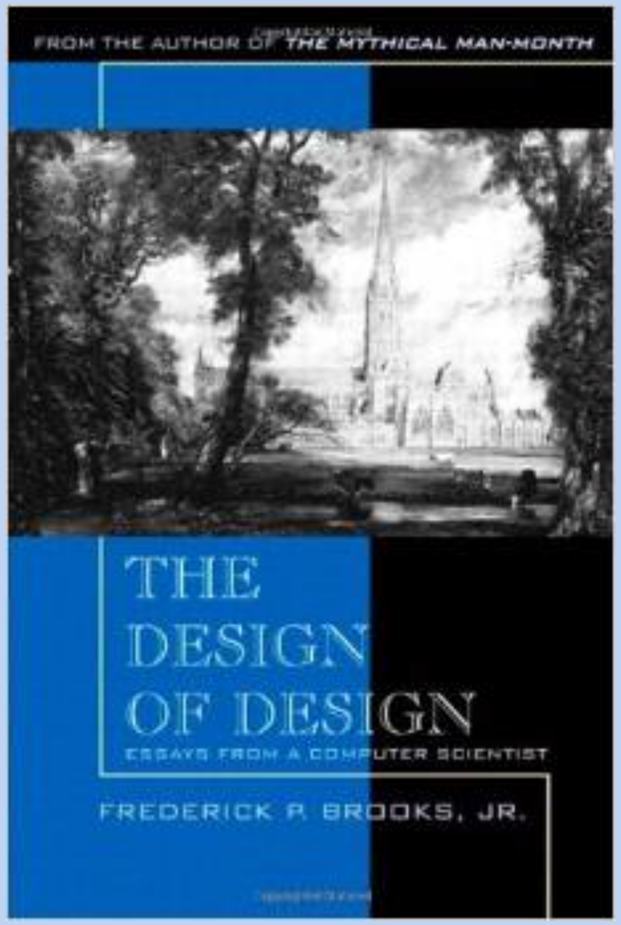
- To make it easy for you to track updates
  - Go to the "What's New" page and
  - Subscribe to the RSS feed
- Feed readers are available for all platforms
  - Feedly, NetNewsWire, etc.
- The website is your source for
  - the class schedule, homework assignments, announcements, etc.
- To turn assignments in and to distribute some class materials, I will make use of D2L, which you can access via MyCUInfo.

Textbooks

Available at the CU bookstore or on-line


The Addison Wesley Signature Series
A Kent Beck Signature Book
User Stories Applied
For Agile Software Development
Mike Cohn
Foreword by Kent Beck


The Pragmatic Programmers
Seven Concurrency Models in Seven Weeks
When Threads Unravel
Paul Butcher
Series editor: Bruce A. Tate
Development editor: Jacquelyn Carter


From the author of The Mythical Man-Month
The Design of Design
Essays from a Computer Scientist
Frederick P. Brooks, Jr.

<http://www.cs.colorado.edu/~kena/classes/5828/f14/textbooks.html>

Three Main Topics

- Agile and User Stories
  - Agile is an example of a software life cycle
  - User stories are the primary way that Agile life cycles capture requirements
- Design and Implementation of Concurrent Systems
  - The days of waiting for faster hardware is (long) gone
  - To make software systems that perform efficiently, you need to incorporate concurrency into your system designs
- Software Design
  - Can you identify/describe the design of a software system? (As distinct from a system's features and requirements?) How does a system's design contribute to the success of a software development project?
    - Contrast Sandvox with Macaw with Muse

Tentative Course Structure

	Dates	Tuesday Topic	Thursday Topic	Theme
Week 1	Aug. 26th, 28th	Course Overview	Intro SE + No Silver Bullet	SE Fundamentals
Week 2	Sep. 2nd, 4th	Intro Agile	Intro Concurrency	Introduction to Topics 1 and 2
Week 3	Sep. 9th, 11th	User Stories 1-3	User Stories 4-7	Agile
Week 4	Sep. 16th, 18th	Concurrency 2	Concurrency 2	Concurrency
Week 5	Sep. 23rd, 25th	User Stories 8-11	User Stories 8-11	Agile
Week 6	Sep. 30th, Oct. 2nd	Concurrency 3	Concurrency 4	Concurrency
Week 7	Oct. 7th, 9th	TBA	Review for Midterm	Review
Week 8	Oct. 14th, 16th	Midterm	Review of Midterm	Midterm
Week 9	Oct. 21st, 23rd	Intro Design	Design 1-5	Introduction to Topic 3
Week 10	Oct. 28th, 30th	User Stories 12-16	User Stories 12-16	Agile
Week 11	Nov. 4th, 6th	Concurrency 5	Concurrency 7	Concurrency
Week 12	Nov. 11th, 13th	Design 6-16	Design 6-16	Design
Week 13	Nov. 18th, 20th	GCD	Concurrency 8	Concurrency
Fall Break	Nov. 25th, 27th	Sleeping	Eating, then Sleeping	Sleeping
Week 14	Dec. 2nd, 4th	Agile 17-21	Design TBA	Agile/Design
Week 15	Dec. 9th, 11th	TBA	TBA	Wrap-Up

Emphasis on Tentative

- The schedule on the previous slide **WILL** change
- However, you can trust that
  - There will be homeworks, quizzes, a midterm, a presentation, and a project
  - **The midterm will be held on Tuesday, October 14th**
    - ☞ CAETE students will need to work with CAETE to identify a person to proctor their midterm exam; You will have from October 14th to October 21st (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 (10%)
  - Homeworks (25%)
  - Midterm (20%)
  - Presentation (20%)
  - Project (20%)
- Quizzes will be taken on D2L; the presentation and project will be submitted on D2L as well

Honor Code

- You are allowed to work together in teams of up to 4 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 October 14th, 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

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/f14/syllabus-statements.html> for more details

Programming Languages

- Code examples this semester will be drawn from a number of languages
  - Java, Objective-C, Clojure, Elixir, C, Ruby, Python, possibly more!
- In general, I'm agnostic on programming languages used for assignments
  - However, some of your homework assignments will require a specific language in order to make use of a specific concurrency framework
    - Take a look at your concurrency textbook to get an idea of the range of languages and frameworks we'll be looking at

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 story on Slashdot from 2012:
    - 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 a 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 a 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 / DevOps
- 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, Homework 1 (Due next Tuesday)
- Lecture 3: Introduction to Software Life Cycles and Agile, Homework 2
- Lecture 4: Introduction to Concurrent Software Systems