

Course Overview

CSCI 5828: Foundations of Software Engineering
Lecture 01 — 08/23/2015

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

David Wheeler



Elliot Loh

@Loh

Follow

Always wanted to travel back in time to try
fighting a younger version of yourself?
Software development is the career for you!

RETWEETS

7,077

FAVORITES

3,918



4:51 PM - 12 Dec 2013

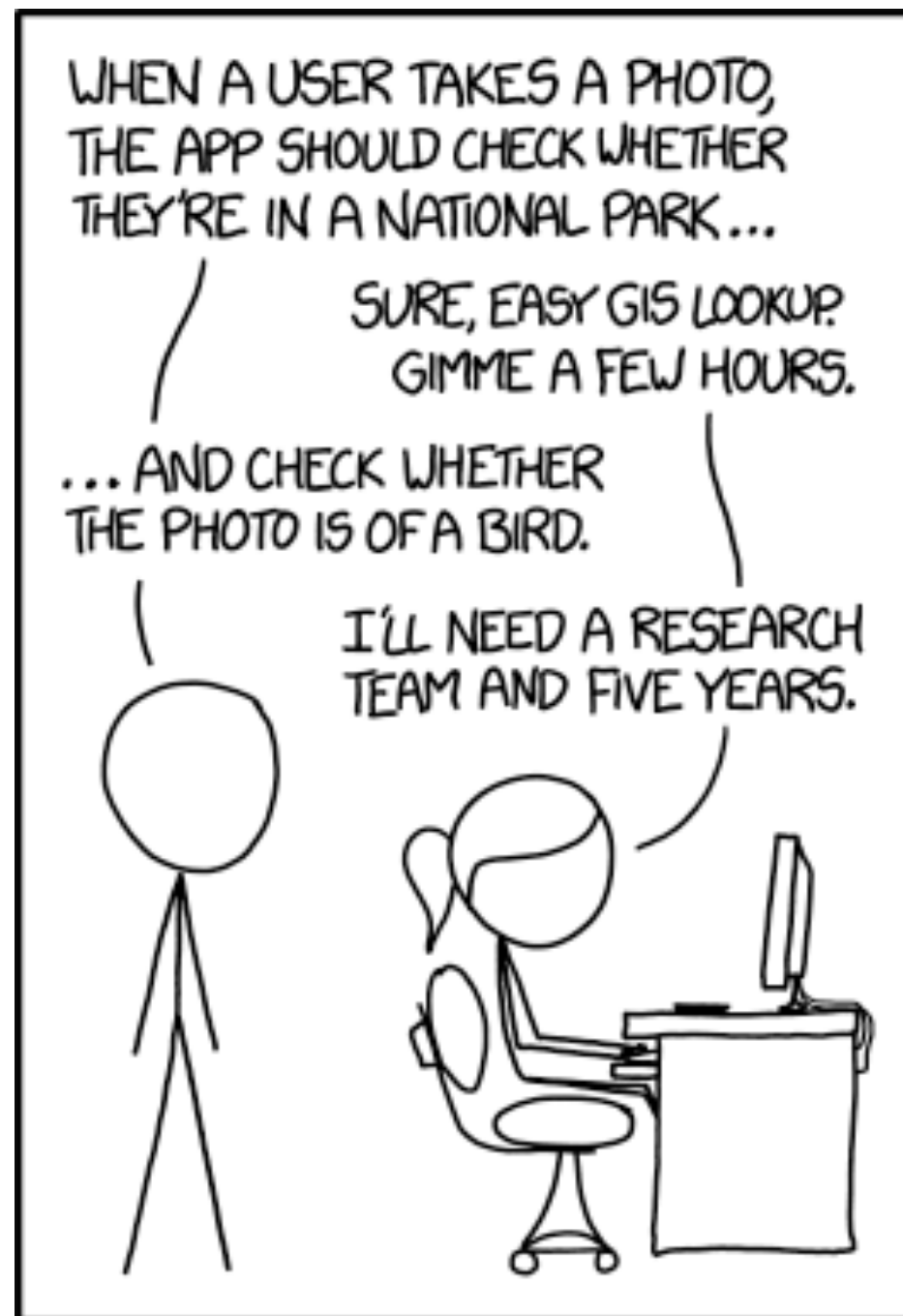
Software can change the world (rapidly)!



Software can change the world (rapidly)!



Software Engineering Can Be Difficult!



IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

Software Engineering is Important



If we fail to our job well, people are negatively impacted

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 two important software engineering concepts
 - designing and implementing concurrent software systems
 - designing and implementing serverless single-page web applications
- with your help (more on that later) explore a wide range of software engineering techniques and tools, software frameworks, and more

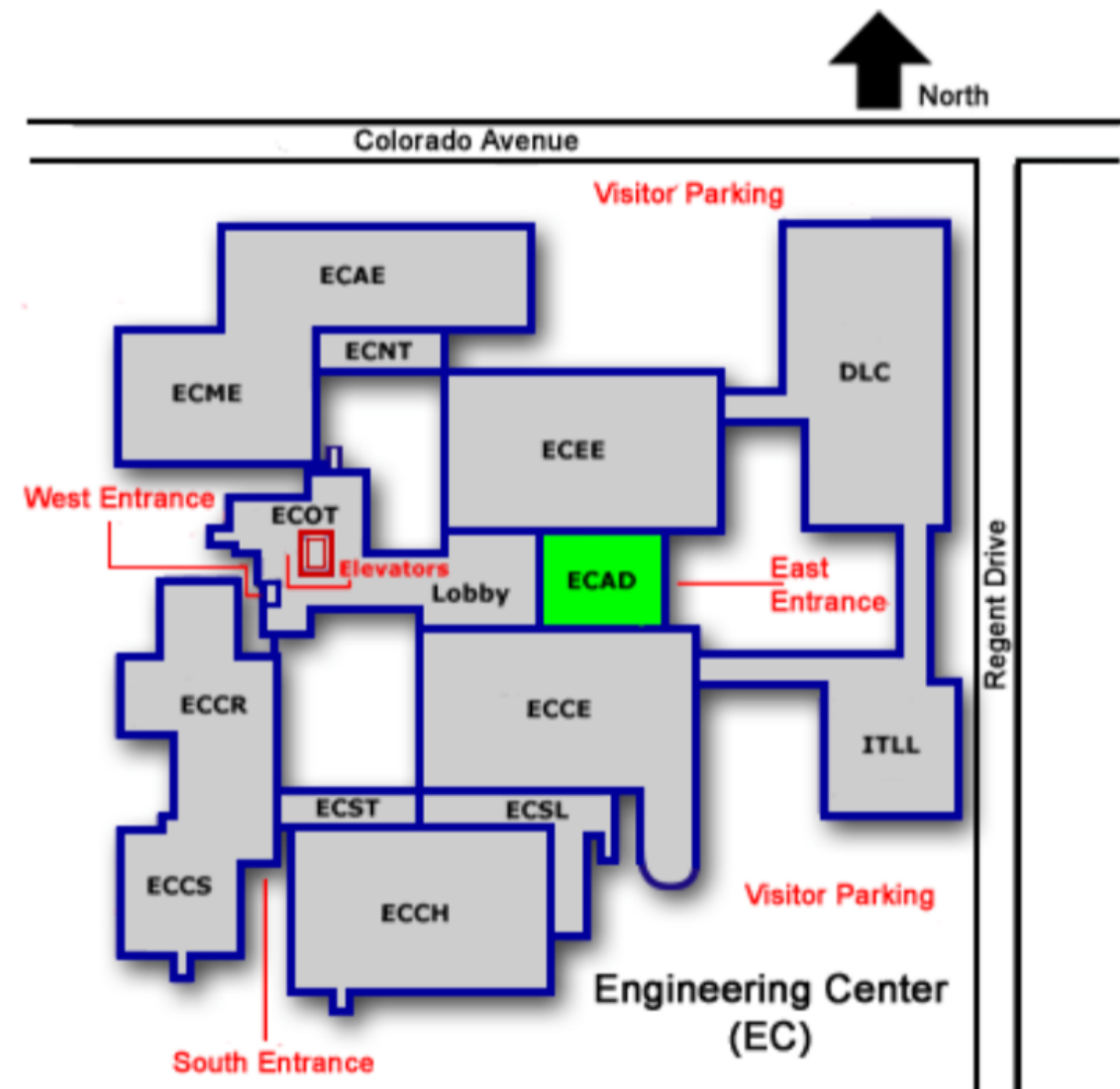
About Me

- Associate Professor
 - Ph.D. at UC Irvine
 - 18 Years at CU;
 - Start of my 37th Semester!
- 10th time teaching this class
- Research Interests
 - Software & Web Engineering
 - Software Architecture
 - Crisis Informatics

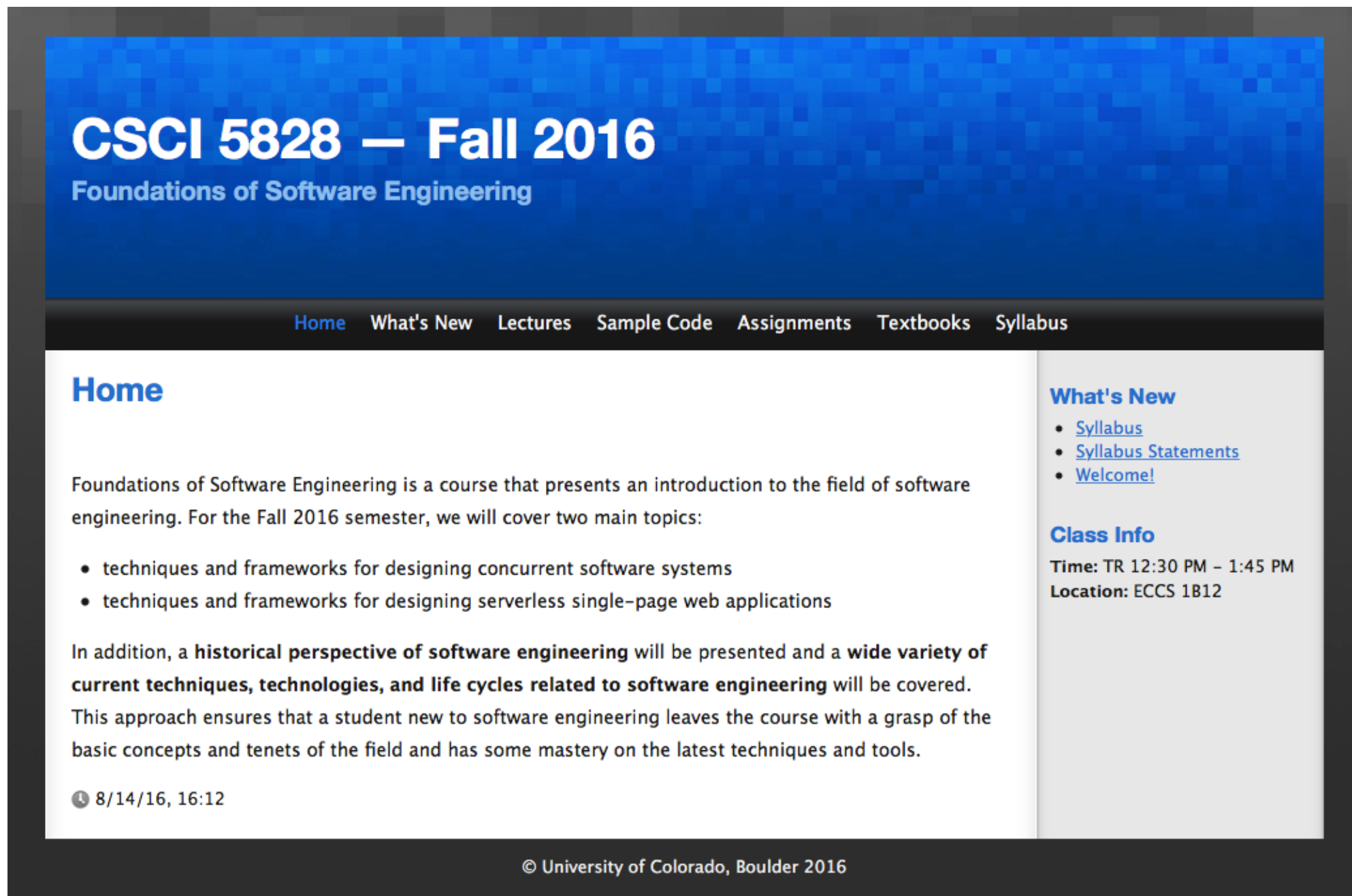


Office Hours

- By appointment only
- Please send me e-mail to schedule a time to stop by
- My office is located in the Dean's Office on the first floor of the Engineering Center
 - The room number is ECAD 101



Class Website



<<http://www.cs.colorado.edu/~kena/classes/5828/f16/>>

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

- 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, and GitHub.
- If you haven't done so, send me your GitHub user id.
 - The repo that we'll be using on GitHub is private
 - For you to access it, I need to add you as a collaborator

Need to learn Git and GitHub? Learn from the Best!



August 31, 6:30-9:30 pm

GitHub's Boulder Office, 1300 Walnut St, Suite 101, Boulder, CO 80302

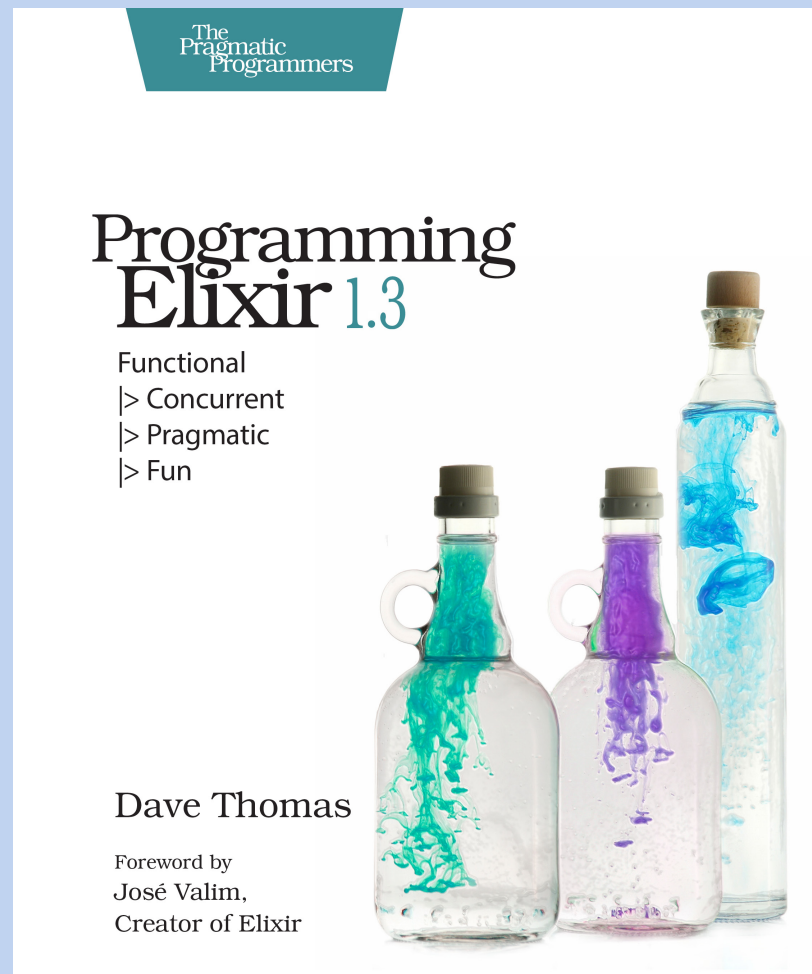
<https://github.com/blog/2227-patchwork-boulder>

Warning! Syllabus is Preliminary

- Why?
 - I don't know how long it will take me to cover a given set of material
 - In some lectures, I'll run out of time and need to pick up where I left off in the next session
 - As a result, a given lecture may cover multiple topics or only part of one
- Be flexible and go with the flow!

Textbooks

Available at <<https://pragprog.com>>



<<http://www.cs.colorado.edu/~kena/classes/5828/f16/textbooks.html>>

Three Main Topics

- Introduction to Software Engineering
 - Overview, history, concepts, techniques, etc.
- 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
- Design and Implementation of Serverless Single Page Web Apps

- PLUS: As a class, we will be exploring lots of software engineering techniques, tools, methodologies, etc.

Course Evaluation

- Your grade will be determined by your work on
 - Homeworks and Semester Project (40%)
 - Essays (40%)
 - Quizzes (20%)
- Semester Project will be integrated into the Homeworks
- Essays will be submitted via GitHub
- Quizzes will be taken on D2L

Essays

- As a class, we are going to perform an in-depth exploration of software engineering topics. You are going to be creating files within our Github repo that provide notes, code, information, and examples of
 - software engineering methodologies
 - software engineering techniques, tools, and frameworks
 - programming languages
 - etc.
- You will be reviewing and critiquing the work of your peers throughout the semester, creating issues and/or pull requests
- You will create **thirteen** of these presentations, during weeks 2-14.
 - More information about this on Thursday

Honor Code

- You are allowed to work together in teams of 4 to 5 people on
 - the homeworks
- The quizzes and essays 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 up to one week late
 - after that the submission will not be graded and you'll receive 0 points for it
- The quizzes and the essays may not be submitted late
 - With respect to the essays, there are two parts
 - creation of an essay
 - review of other student essays
 - if you're late with your essay, you will impact the work of other students

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/f16/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!
- Within assignments this semester, we'll be making heavy use of
 - Elixir
 - Javascript
 - HTML
 - CSS
- We'll be reviewing these technologies throughout the semester

What do you know about Software Engineering?

- Fire up your web browsers and head to
 - <http://bit.ly/2bFkR75>
- Let's spend some time finding out what people in the class know about software engineering
 - then we'll look at some of my definitions and perspectives

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**

Our primary tool?

- Abstractions
 - When it comes down to it, software engineers solve problems by
 - **developing abstractions** that break the problem down into something that is understandable
 - and/or
 - by **using abstractions** developed by others
- The file system? An abstraction. A database? An abstraction. Twitter's API? An abstraction. Your own Employee class? An abstraction.
 - It's abstractions all the way down...

Example Abstractions

- What does this abstraction let you do?
 - Q: $^{\wedge}(.^*?), (.^*?)^{\$}$
 - R: $\setminus 2 \setminus 1$
- How about this one?
 - `history | grep brew`
 - `find . -type f -name *.scala -print | wc -l`
- How would you construct a social graph of two hops for a given set of user ids?
 - `get_friends.rb`: takes list of ids and provides list of ids *followed* by them
 - `get_followers.rb`: takes list of ids and provides list of ids that *follow* them
 - `get_user_objects.rb`: takes list of ids and provides user information for each one

Emphasizing the Point: Conversation is Key

- How do we understand the problem we're trying to solve?
 - Conversations with users and domain experts
- How do we understand an abstraction that someone else wrote?
 - Conversations on-line with other developers or “with” the documentation
- How do we understand what abstraction we should write?
 - Design Conversations
- How do we know if our abstraction is working?
 - Testing (“Conversations with test cases”)
- etc.

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**
 - The paper is available on our private GitHub repo for the class
 - (Yet another reason why I need your GitHub user id!)

Questions?



Coming Up Next

- Lecture 2
 - No Silver Bullet
 - Git and GitHub
 - Homework 1 Assigned (Due next Tuesday)