Course Overview

CSCI 5828: Foundations of Software Engineering
Lecture 01 — 08/25/2015
All problems in computer science can be solved by another level of indirection.

David Wheeler
Always wanted to travel back in time to try fighting a younger version of yourself? Software development is the career for you!
Almost ready to release highway autosteer and parallel autopark software update
Final corner case is dealing with low contrast lane markings (faded white on grey concrete) while driving into the sun at dusk.
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
  • software development life cycles, with an emphasis on agile methods
  • designing and implementing concurrent software systems

• 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
  • 17 Years at CU;
    • Start of my 35th Semester!
• 9th 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 2015
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 2015 semester, we will cover two main topics:

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

In addition, a historical perspective of software engineering will be presented and a wide variety of current techniques and technologies related to software engineering will be covered. This approach 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.

8/24/15, 14:41

What's New
- Welcome!

Class Info
Time: TR 12:30 PM - 1:45 PM
Location: ECCS 1B12

<http://www.cs.colorado.edu/~kena/classes/5828/f15/>
Check the website every day! (I’m serious)

• If you want, the “What’s New” page has an RSS feed
  • you can subscribe to that feed to be notified when new posts arrive

• 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 <https://pragprog.com>

<http://www.cs.colorado.edu/~kena/classes/5828/f15/textbooks.html>
Three Main Topics

• Introduction to Software Engineering
  • Overview, history, concepts, techniques, etc.

• Agile Development
  • Agile is an approach to software development
    • It has many life cycles (Extreme Programming, Kaban, Scrum, 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

• 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
  • Class Participation and Attendance (5%)
  • Quizzes (10%)
  • Homeworks (35%)
  • Midterm (20%)
  • Presentations (30%)
• Quizzes will be taken on D2L; the presentations will be submitted on GitHub
Midterm

• The midterm will be held on Tuesday, October 13th

  • BBA students will need to work with BBA to identify a person to proctor their midterm exam; You will have from October 13th to October 20th (one week) to take your exam and have it sent to me by your proctor
Presentations

- As a class, we are going to perform an in-depth exploration of software engineering topics. You are going to be creating Github repos 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 five of these presentations, one every three weeks
  - I will present an example of what I’m looking for on Thursday
Honor Code

• You are allowed to work together in teams of up to 2 people on
  • the homeworks
  • the presentations
• 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 presentations 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 midterm may not be submitted late
    • If you discover that you cannot attend the midterm on October 13th, 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/f15/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 do you know about Software Engineering?

• Fire up your web browsers and head to
  • http://bit.ly/1NQcQIq

• 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
    - It’s abstractions all the way down…
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
    • Paper is available on IEEE Digital Library: No Silver Bullet
    • It is also available on the D2L website for this class
Questions?
Coming Up Next

• Lecture 2: No Silver Bullet, Homework 1 (Due next Tuesday)