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

<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

<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

© Kenneth M. Anderson, 2014
## Tentative Course Structure

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

© Kenneth M. Anderson, 2014
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 \( \Rightarrow \) i.e. maintenance headaches
      • plus, we’re just duplicating (badly) the functionality that more powerful
          tools can provide

• 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