Course Overview

15-413: Introduction to Software Engineering

Jonathan Aldrich

How is SE in industry different from coding assignments?

- *Student comments*
  - *Boss has no idea what he wants*
  - *Spend more time on testing than you ever dreamed*
  - *Have to maintain code weeks/months/years*
  - *Adapt your code to bigger infrastructure*
  - *Unsure what correct output*
  - *Documentation requirements more rigorous*

How is SE in industry different from coding assignments?

- Some of my answers:
  - Requirements ambiguous
  - Requirements change during development
  - Scale is larger
    - Requires different design skills
    - Requires teamwork
  - Software must be changed after development is complete
  - Failure is more expensive
    - Business-critical
    - Safety-critical

Assignment 1

- Read a software engineering case study
  - Several options, from SE Ph.D. students' experiences
- Write your reaction to the study
  - What did you find surprising?
  - Was there anything you could relate to your own experience?
  - 1 page (at least 500 words)
- Due Friday

What is Software Engineering?

- Involves whole development cycle
- Implementation of a process that guarantees good results
- Break down problems and solve them
  - Test, revise and try
  - Design, create, test, iterate
  - Design for errors & compensate

What is Software Engineering?

- One definition (Mary Shaw)

Software Engineering is

- *the branch of computer science*
- *that creates practical, cost-effective solutions to computing and information processing problems,*
- *preferentially by applying scientific knowledge*
- *developing software systems in the service of mankind.*

How does Software differ from other engineering disciplines?

- *Student comments*
  - *Newer discipline*
  - *Easier to revisions*
  - *Innovation/pace of change*
  - *Not physical – more ways it can break*
    - *Laws underneath are more complex*
  - *Can have many purposes, and can change*
  - *Not as much time spent testing*
    - *Not required to be as robust*
  - *Management difficult because hard to measure quality/intangible*

How does Software differ from other engineering disciplines?

- Some of my answers:
  - Software is designed, not manufactured
    - Production cost is paid up front
    - Little re-use achieved in practice
  - Software is based on discrete math
    - Butterfly effect: small errors can have big consequences
    - Overengineering does not work well
  - Software is malleable
    - Can apply to huge variety of problems
  - Software doesn't wear out
    - All problems are "designed in"

Course Goals

- You will leave the course:
  - **Understanding the role of software in systems**
    - How software differs from other engineering materials

Course Goals

- You will leave the course:
  - Understanding the role of software in systems
  - **Understanding why SE practices are important**
    - Reading and analyzing historical SE failures
    - Being exposed to situations that require good SE practices
    - Using SE practices enough to see value in them
    - Reflecting on influence of SE practices in course project

Course Goals

- You will leave the course:
  - Understanding the role of software in systems
  - Understanding why SE practices are important
  - **Knowing good basic SE practices**
    - Software process and project management techniques
    - Requirements elicitation
    - Design and Architecture techniques
    - Coding best practices
    - Testing and analysis of code

Course Goals

- You will leave the course:
  - Understanding the role of software in systems
  - Understanding why SE practices are important
  - Knowing good basic SE practices
  - **Able to make simple engineering tradeoffs**
    - Exposure to multiple techniques with benefits/drawbacks
    - Making decisions in practice and reflecting on consequences
    - Evaluation of tradeoffs in historical SE projects and in peer class projects

Course Goals

- You will leave the course:
  - Understanding the role of software in systems
  - Understanding why SE practices are important
  - Knowing good basic SE practices
  - Able to make simple engineering tradeoffs
  - **Possessing basic skills using SE tools and practices**
    - Exposure to tools: Debuggers, version control, configuration management, unit tests, modeling tools, analysis tools
    - Skills for working within frameworks and large systems
    - Ability to carry out basic software life-

Course Goals

- You will leave the course:
  - Understanding the role of software in systems
  - Understanding why SE practices are important
  - Knowing good basic SE practices
  - Able to make simple engineering tradeoffs
  - Possessing basic skills using SE tools and practices
  - **Having applied those skills in a structured setting with realistic challenges**
    - CMU philosophy: include application as well as fundamentals
    - Structured to frame an SE context

Course Emphasis

- Technical content
  - Design
  - Analysis
  - Quality assurance
- Management
  - Teamwork
  - Working for clients
  - Project Planning
- Experience
  - Real project for a CMU client
  - Homework exercises

Project

- Real, internal CMU client
  - Provides interesting problem, realistic pressures, unclear/changing requirements, etc.
  - Lower overhead and pressure than external client
- Small, 3-4 member teams
- Emphasis on good SE practices
  - Homeworks and deliverables tied to project
  - Grading: practices more important than end result

Evaluation

- Homework
- Project deliverables
- Class presentations
- Client assessment
- 360-degree peer evaluations
  - You will evaluate your team members and yourself

Textbook

- Optional text
  - Roger S. Pressman, Software Engineering, A Practitioner's Approach
- Readings from the literature
- Other resources
  - Brooks, Mythical Man-Month
  - Sommerville, Software Engineering
  - Glass, Software Runaways
  - Design Patterns

Course Outline

- Weeks 1-3: Process, Planning, Estimation, Risk Management
- Week 4: Requirements
- Week 5: Architecture
- Weeks 6-7: Design
- Week 8: Formal Methods
- Week 9: Coding
- Week 10-11: Quality Assurance
- Week 12: Analysis
- Week 13: Responsibilities of an Engineer
- Week 14: Software Evolution
- Week 15: process improvement, wrapup

Project Outline

- Week 1: Form teams & bid for project
- Weeks 2-3: Planning, Requirements
- Weeks 4-6: Requirements, Prototyping
- Week 7: Architecture
- Week 8: Design
- Week 9-10: Formal modeling assignment
- Week 11: Test Plan
- Week 12: Code review assignment
- Week 13-14: Analysis assignment
- Week 15/Finals: Final Report

A reminder on plagiarism

- Do not copy material (code, homework) without attribution
  - Plagiarism is cheating; the minimum penalty will be a zero for the assignment
- Your work should be your own
- If you have any questions, ask the instructor or a TA

Questions?