ECE 458
Engineering Software for Maintainability

Intro
Welcome!

• Welcome to ECE 458:
  • Engineering Software for Maintainability
  • Your Senior Design Course!

• Quick introductions:
  • Please feel free to just call me Drew
  • ...and I’d like you all to introduce yourselves
What this class is about

• Real software has a long lifespan
  • In industry, you might work the same code base for years or decades

• Contrast with code you write in school:
  • Turn it in, forget about it.

• Real world software’s requirements evolve
  • New features
  • Changing requirements
  • ...

• How do we design software to ease later changes?
  • Goal of this class: learn this by doing and reflection
What this class is not

- This class is not about learning to program
  - I assume you already are a competent programmer
    - You must have taken CS201, CS 250, CS 308 to be here..

- This is not a lecture class
  - Today is the only time I will have prepared slides
  - I could talk about software design all day, but...
    - You will really only learn by doing
    - “I’m not the sage on the stage, I’m the guide on the side”
    - So many profs say that, but in this case, I mean it.
What are we doing?

- One semester long project:
  - Resource (room, projector,...) Scheduling Software
    - Feedback from prior year: “do something other than a game.”
  - Requirements staged into 4 evolutions.
    - Add or change requirements from prior evolutions
- With each evolutions, submit a writeup. Two major parts:
  - Forward looking (analysis of current design):
    - What are its key features?
      - Why did you design it this way?
    - What do you see as its strengths?
    - How about its weaknesses?
  - Retrospective (analysis of past design choices):
    - How did your past designs set you up to win or struggle?
    - How did these outcomes align with your prior analyses?
Project Groups

• You will do your project in groups of 4
  • Pick carefully: fixed for the semester

• Considerations:
  • Language choices
    • Note: subject of next Monday’s discussion
  • Other tool choices
    • Revision control,…
  • Skills and expertise
    • Ideal: strong skills, complimentary expertise

• End of class: find groups, start planning ev1
Writeups

- No specific page limit/requirement
  - Say what you need to say. Don’t say more, don’t say less.
- Highly recommend LaTeX + version control (git,...)
  - You are all engineers: make good use of your tools
  - Submit pdfs/LaTeX only (no Word docs)
- Expect document to be
  - Well-written:
    - Organized, clear, precise.
    - Include figures if they help
  - Analytical:
    - Delve into why your design is/was good/bad
    - Tell me what was bad, and how it could have been better
      - Hindsight is 20/20
    - Include discussion of testing plan (part of design)
Oral Presentations

- Day that evolutions are due: oral presentations
  - Each group members presents once
- 17 minutes per group
  - 75 minutes / 4 groups = 18.75 min
  - but need some time to change groups, setup, etc.
- Rough outline (~4 min each)
  - Quick demo of working project
  - Retrospective from previous evolution
  - Overview of current design
  - Analysis of current design (include: why, strengths, weaknesses)
- Tight timeline, but don’t rush
  - 4 minutes is actually a pretty long time
Project Deadlines

• Evolution 1: Before class on 2/10  [26 days]
  • Released (now)
• Evolution 2: Before class on 3/2  [21 days]
• Evolution 3: Before class on 3/30  [21 days + spring break]
• Evolution 4: Before class on 4/22  [23 days]
Class Time: Two ways

- Class discussions:
  - Topics posted on class webpage (all posted now)
    - Feel free to prepare other discussion points than those listed
  - Readings (short)
    - Some TBD
    - Requested by previous classes
  - Prepare ~1--2 pages of outline/notes on discussion
    - Preparation should take ~30 minutes per discussion
    - Typed up, will hand in during class
  - Some discussions: general topics (good design, documentation..)

- Workdays
  - Work with your group on your project
  - I’ll circulate around, answer questions, offer advice, etc.
Grading

• 45% software deliverables:
  • How well did your code work? How well was it designed?

• 25% written deliverables:
  • Technical/analytical content: how well did you describe/analyze?
  • Writing: how well written are your documents?
    • More strict as semester progresses

• 10% oral presentation:
  • Each group member does one evolution’s presentation

• 20% class attendance/participation:
  • Come to class regularly (2 free absences).
  • Have your discussion notes prepared (grading: 0, 70, or 100)
  • Actively participate in the discussions

• No exams
Academic Integrity

• Expect academic integrity from all of you
  • Duke community standard
    • I will not lie, cheat, or steal in my academic endeavors, nor will I accept the actions of those who do
    • I will conduct myself responsibly and honorably in all my activities as a Duke student.

• Concrete rules:
  • Discuss anything you want
    • Give credit where it's due if you use other groups’ ideas
  • All code should be produced within your group
    • Don’t share code outside your group
    • Can use libraries for graphics, sound, etc (e.g., SDL) as needed

• Not sure? Ask
Specifics of the Project

- Project: Resource Scheduling Software

- Many specifics are left up to you
  - Web based? Desktop application? Mobile app? Your choice

- All 4 evolutions are already written
  - I won’t change them, no matter what you say in discussions
    - Some students worried that I might put in their worst fears
Requirements (Continued)

• Requirements will be distributed as pdfs
  • Colors to show new/changed requirements
  • P.S. one LaTeX source, macros to control evolutions

• Unclear on requirements? Ask
  • Happy to clarify anything

• Unspecified requirements/behavior?
  • Do anything reasonable

• Don’t need to be artistic
  • Though feel free to make it look nice if you want
Submission

• Submission of projects by repository pull
  • Your choice of revision control system
  • Include writeup in repository

• Server:
  • Have a test server and a production server
  • Production server should have working evolution for 2 weeks after due date
  • Recommend VM from OIT: http://vm-manage.oit.duke.edu

• Please nothing that requires Windows
  • Linux or Mac are fine
  • If you do anything mobile, please include instructions for emulator
Questions?

• Any questions?
  • Before I turn you loose on evolution 1...
Evolution 1: Go!

- You all have a copy of the Evolution 1 requirements
  - Time to get started!
- Find your groups
- Start talking about your design
  - Sketch out some UML?
  - Decide how to split up the work?
  - What do you think the main challenges will be?
  - How should you design to accommodate whatever changes I throw at you?
  - What programming language do you want to use?
    - Detailed discussion on Wednesday.
- Also, a good chance to ask for clarifications on the reqs