ECE 458
Engineering Software for Maintainability

Introduction and Course Overview

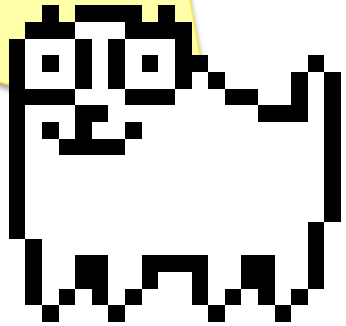
Tyler Bletsch

(Adapted from work by Drew Hilton)

Welcome!

Welcome to ECE 458:
Engineering Software for Maintainability

Your Senior Design Course!



Evolution!

- You four years ago:

Fig. 1: Freshman year

- You now:

Fig. 2: You now have freaky robot hands I guess??

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
  - ...

Fig. 3: Software is like pokeymans

- 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, you know that (well, you better know that...)
- This is not a lecture class
  - These are the first, last, and only slides I've prepared
  - You've been taught some software engineering skills, but...
- **You learn by doing!**

What are we doing?

- One semester long project:
  - An inventory and checkout system for the ECE department
    - YOUR SOFTWARE, IF CHOSEN, WILL ACTUALLY GET USED BY PEOPLE!!
  - Requirements staged into **4 evolutions**.
    - Changes will usually be substantive restructuring of core ideas: Not "add this form", but "change what this concept means"
- After each evolution, submit a report. Major parts:
  - 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?
  - 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?

Where's all the stuff?????

It's here:

http://people.duke.edu/~tkb13/courses/ece458/

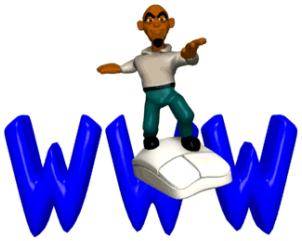
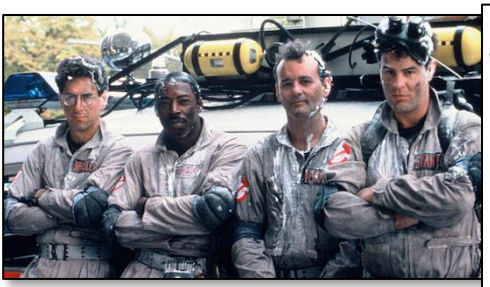


Fig. 4: internet.

Project Groups

- You will do your project in groups of 4
  - Pick carefully: fixed for the semester
- Considerations:
  - Language/framework choices
    - Note: subject of next discussion
  - Other tool choices
    - Revision control, …
  - Skills and expertise
    - Ideal: strong skills, complimentary expertise
- End of class: find groups, start planning ev1

Fig. 5: All the best groups have four members




Project reports

- No specific page limit/requirement
  - Say what you need to say. Don't say more, don't say less.
- 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)

Fig. 6a: WikiHow illustrations will never stop being hilarious

Oral Presentations

- Day that evolutions are due: oral presentations
  - Each group members presents once
- 10 minutes per group
  - The *seemingly least prepared* group goes first!
- Rough outline (2—3 min each)
  - Quick demo of working project
  - Retrospective from previous evolution
  - Overview of current design
  - Analysis of current design (include: why, strengths, weaknesses)

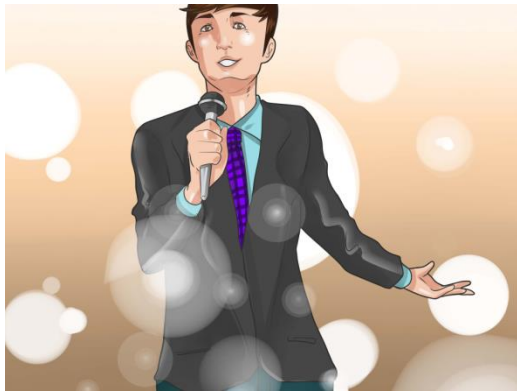


Fig. 6b: See?

Class Time: Four flavors

1. Class discussions:
   - Topics posted on class webpage (all posted now)
   - Some topics have readings – you need to read it before you come
   - Prepare ~1-2 pages of outline/notes on discussion, turn in at end
     - The notes are NOT a summary of the reading, but your thinking on the whole of the topic (reactions, opinions, questions, etc.).
   - Two **discussion leads** per topic (sign up soon!)
2. Workdays
   - Work with your group on your project
   - I'll circulate around, answer questions, offer advice, etc.
3. Presentation day
   - If presenting: present. Else: support your group!
4. Reflection day
   - Session after a due date
   - Reflect on work so far, discuss newly released requirements

Ask questions, please!

- Discussions are a great place to ask questions!
- Last time, students reported that they felt intimidated to ask about concepts that were introduced in discussion
  - But it turns out it was MOST students who felt this way!

- **Imposter Syndrome:** The phenomenon wherein "high-achieving individuals are marked by an inability to internalize their accomplishments and a persistent fear of being exposed as a 'fraud'. "

- Affects 90% of Duke I think... (including myself)

Fig. 7: Human psychology

Stand-up meetings

- A brief meeting where teams go over status, blockers, and open questions.
- Every Tuesday except for presentation days
- Noted on the class schedule

Fig. 8: Hahaha look at this terrifying menagerie of blob people

Grading (1)

- 45% software deliverables:
  - How well did your system meet requirements?
  - Based on a **demo session** and **instructor functional testing**
  - We want to be as objective as possible, but in assessing "quality" without the benefit of a giant spec doc, there will be some subjectivity.
    - The system must actually be *good* from a customer perspective, not merely tick all the boxes.
    - Especially true for problems reported to you that you do not fix!
- 25% written deliverables:
  - Technical/analytical content: how well did you describe/analyze?
  - Writing: how well written are your documents?

Grading (2)

- 10% oral presentation:
  - Each group member does one evolution's presentation
  - Rubric will be posted
- 20% class attendance/participation:
  - Come to class regularly (2 free absences).
  - For discussion:
    - Have your discussion notes prepared (grading: 0, 70, or 100)
    - Actively participate in the discussions
  - For workdays/presentations/reflections:
    - Attend and participate as appropriate
- No exams! 

Advice from past students

Fig. 9: Dr. E. L. Brown, ca. 2015

From a survey given at the end of Spring 2016. Unedited.

- "Make sure that you set individual milestones during an evolution, this is one of the classes where actually trying to do everything the weekend before will not work regardless of how smart you think you are."
- "Take a long time deciding which framework, database and front end to use as it really affects future evolutions. Really look at the data and the requirements in your design decisions (especially databases), rather than going with just what you know or a 'shiny new tool'."
- "Web Development"
- "Don't switch frameworks at any point during the project. It is a fucking nightmare. Not worth it. Beware picking a framework that is cool and trendy that you know nothing about...don't do it "just to learn" - make sure it's got good documentation and support."
- "Understand that this is a class about satisfying software requirements. This is not a class about writing modular, clean code. Yes, doing that will make it easier to adapt to changes in the specifications, but it will not be a part of your grade and no one will look at your code."
- "Anything to do with web development, unfortunately we have none of those at Duke."

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 its 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: ECE Inventory System

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

- All 4 evolutions are already written
  - I will not change them in response to your status

    (Some students worried that I'd put in their worst fears)
  - However, I may change them in response to feedback from the ECE department lab staff (i.e., the *real* customer)

Requirements

- Requirements will be distributed as PDFs
  - New requirements in blue
  - Changed requirements have old requirements in grey
    - (replacements in blue)

- Unclear on requirements? Ask
  - Happy to clarify anything
- Unspecified requirements/behavior?
  - Do anything reasonable
- Contradiction? Is a requirement stupid?
  - Discuss it in class or on the forum; changes may be possible
- Don’t need to be artistic
  - But it does need to be efficient and usable!

Submission

- Submission of projects by repository pull
  - Your choice of revision control system
  - Coordinate with TA on submission
- Server:
  - Have *at least* a test server and a production server
  - Production server should have working evolution for 2 weeks after due date
    - We test on this deployment when you're not around!
  - Recommend VM from OIT: http://vm-manage.oit.duke.edu
  - NOT recommended: the various fly-by-night "free" hosting providers
    - Students have been screwed by this before...
- A note on platform:
  - You must document ALL environmental pre-requisites and instructions for setup of your product
  - If you do anything mobile, please include instructions for emulator

QUESTIONS?

A realistic beginning

- The formal requirements have NOT been published yet
  - They will be posted soon

- To get started, you'll need to interview the customer:
  - The Duke ECE department's laboratory staff (as impersonated by me)

Fig. 10: Release planning

Minor logistics

- For next time, need to select:
  - Four to present a programming language + framework
  - Two to act as discussion leads for the programming discussion

Evolution 1: Go!

- Find your groups
- Start trying to get the requirements out of the customer (me)
- Maybe even talk 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 Monday.
  - What procedures and tools to use?
    - Code control? Scheduling? Milestones? Task tracking? Etc.