SWE 265P
Reverse Engineering and Modeling

Lecture 1

Reality

"One thing that was counterintuitive is that in my first 'real' software engineering jobs, I spent 80-90% of my coding time reading and 10% writing." – Eric Dashofy [General Manager & Deputy CIO, The Aerospace Corporation]

Today

- Logistics
- Nature of the course
- Why reverse engineering
- Setting up your environment
- First homework

Logistics

- Professor: André van der Hoek (andre@uci.edu)
- TA: Kaj Dreef (kdreef@uci.edu)
- https://www.ics.uci.edu/~andre/swe265pw2020.html

Logistics

- Office hours
  - every Tuesday, 09:00-10:00
  - by appointment
- Open door policy
  - DBH 5038
- Slack
  - SWE-265P-W2020
- TA will have office hours every Wednesday, 10:00-11:00, and you can also reach him via e-mail

Goals

- For you to be able to effectively use a set of strategies and tools for understanding, working with, and changing large software
- For you to be able to effectively model different aspects of large software
- For you to initiate or expand your professional portfolio by making small contributions to a large open source software system

Secondary goal

- For you to be able to talk intelligently about this course and smartly draw upon its content when you interview

A note on the course title

- SWE 265P – Reading code
- SWE 265P – Software understanding
- SWE 265P – Program comprehension
- SWE 265P – Making sense of source code (at scale)
- SWE 265P – Software models, visualization, and change
- SWE 265P – …

Place in the curriculum

- SWE 241P–248P
  - hands-on software development, focused topics
  - small programs
- SWE 264P
  - theory and practice of software architecture
  - large-scale software
- SWE 261P
  - theory and practice of software quality
  - large-scale software
- SWE 265P
  - theory and practice of reading code
  - large-scale software

Structure of the course

- Lecture
- Practice in class
- Learn from professionals
- Course project
  - five parts
- Midterm & final
- Diary

Grading

- Course project (50%)
  - each part 10%
- Diary (10%)
- Midterm (15%)
- Final (25%)

- *Up to two parts of the homework project can be resubmitted to improve your grade, each within five days of receiving the grade*

Basic tenor of the course

- Weeds
- Hands-on
  - bring laptop to class every lecture
- Trial and error
- No ‘reverse engineering recipe’

Basic tenor of the course

- Conversation
- Challenges
- Mistakes
- Tenacity
- Work

Working together, working alone

- All work on your project shall be performed with your team only
- The midterm and the final shall be performed by yourself only
- Failure to do so will be considered a violation of academic integrity
  - aisc.uci.edu

Questions?

My question #1

- What are some reasons that we need to read and comprehend source code?

Answer #1

- Learning to read before learning to write
- Understanding the functionality that the software has
  - what it already does
  - what it does not do
- Understanding how the code realizes its functionality before changing it
- Compensating for documentation that might not be accurate/up-to-date
- Reviewing code (pull requests) as part of organizational ethos
- Programming/debugging with a partner

Answer #1 (continued)

- Finding help on StackOverflow (or other fora) for a programming issue being faced
- Looking for how another software system might have solved a given programming problem
  - same programming language
  - other programming language
- Expanding repertoire of design/coding solutions for personal growth
- Assessing components, libraries, and services before adopting them (or not)
- Forgetting the exact details of and/or rationale for an old piece of code

My question #2

- What is the largest piece of software you have worked on?
  - what did it do?
  - what specific task or tasks did you work on?

My question #3

- Give examples of when it was easy to read, understand, and perhaps modify a piece code
  - why was it easy?

My question #4

- Give examples of when it was difficult to read, understand, and perhaps modify a piece code
  - why was it difficult?

Answer #3/#4: factors affecting program readability

- Reader characteristics
- Intrinsic factors
- Representational factors
- Typographic factors
- Environmental factors

Break

Theory and practice

- Little theory exists on how to ”best” read software
- Many tools exist, but each has its (significant) limitations
  - functionality
  - correctness
  - no rationale
- Much, ultimately, is rooted in practice and as a result of such practice building an understanding of one’s own patterns of what works and what does not work, when
- Work
  - practice
  - reflection

Setting up your environment

- Java
  - https://www.oracle.com/technetwork/java/javase/downloads/jdk13-downloads-5672538.html
- Git
  - https://git-scm.com/downloads
- Subversion
  - https://www.collab.net/downloads/subversion

Setting up your environment (continued)

- IntelliJ IDEA
  - https://www.jetbrains.com/idea/
- Statistic plug-in
  - <ctrl-alt-s>
  - <plug-ins>
  - <marketplace>
  - <Statistic>
  - <install>

Setting up your environment (continued)

- Download, build, and run:
  https://github.com/githubtraining/hellogitworld.git

- Download, build, and run:
  svn://svn.code.sf.net/p/jedit/svn/jEdit/trunk/

Theory and practice

- Little theory exists on how to ”best” read software
- Numerous tools exist, but each has its (significant) limitations
  - functionality
  - correctness
  - intent
- Much, ultimately, is rooted in practice and as a result of such practice building an understanding of one’s own patterns of what works and what does not work, when
- Work
  - practice
  - reflection

Reflection

- Maintain an ongoing diary
  - date
  - time
  - participants (as applicable)
  - goal
  - accomplishments
  - insights
  - mood

- Submit regularly through GitHub pull requests

Setting up your environment (continued)

- Clone:
  https://github.com/swe-265P/W2020
- Inside the folder *diaries*, copy *diary-template.xlsx* to *diary-<yourname>.xlsx*
- Add today's activity
- Submit pull request

Homework

- Make sure you are familiar with the basics of IntelliJ IDEA
- Download a few Java open source applications, of varying sizes, build them, and run them (minimum 3)
- By midnight of Wednesday, January 15, submit your updated diary as a pull request on GitHub

Homework (continued)

- Form a team of 3, which will be your team for the duration of the quarter
  - one team will be 4

- Create a team directory in swe-265P, add your names to the teams.xlsx spreadsheet, and submit a pull request

A final word

- SWE 265P is a unique course
- SWE 265P is experimental in nature
- SWE 265P will be what you make of it
  - in engaging with the materials
  - in providing feedback on course direction & content