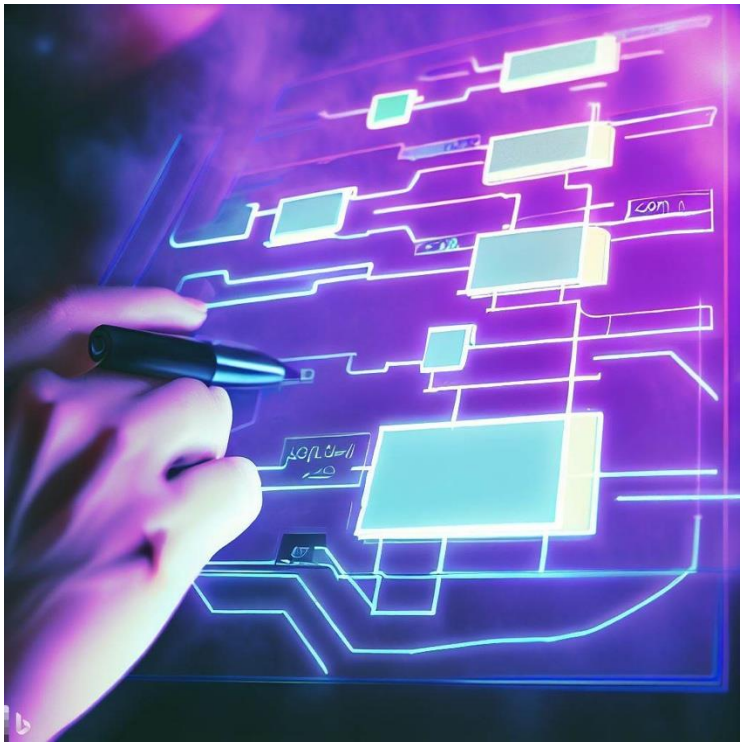


CSE 331

Software Design & Implementation

James Wilcox

About Me

James Wilcox

- Asst Teaching Prof in CSE
  - did my PhD here 2013-2019

- Interested in:
  - programming languages
  - distributed systems
  - teaching programming
  - music (mostly choir)
  - running

About Me

James Wilcox

- Asst Teaching Prof in CSE
  - did my PhD here 2013-2019

- Built a wide range of systems and applications

Systems

- compilers
- operating systems
- distributed systems
- networking systems
- database systems
- graphics
- ...

Applications

- desktop apps
- web apps
- phone apps
- IDE
- games
- ...

About You

- Familiar with Java at the level of 123
  - understand primitive vs reference types
  - familiar with recursive functions
- Probably in your first/second year of the major

About This Class

- **Very little content change vs 10 years ago**
  - **maybe 10% changes based on who teaches it**

- **Substantial homework changes**
  - **made a big change ~3 years ago**

    typical student went from senior to sophomore
  - **making another change this year**

    need to consider the role of AI

"Coding, or the translation of a precise design into software instructions, is dead. AI can do that."

— Magda Balazinska

About This Class

"Coding, or the translation of a precise design into software instructions, is dead. AI can do that."

— Magda Balazinska

- We will not do any coding this quarter (in that sense)
  - we will focus on the parts that come before and after
- Before coding, you must write a *precise* design
- After coding, you must check that it is correct

Before Coding

- A precise design requires a precise "specification"
  - says exactly what input/output behavior is expected
  - AI cannot read your mind!

- A correct specification should:
  - rule out every implementation you don't want
  - rule out no implementation you would accept

- Surprisingly difficult to do

Before Coding

- A precise design requires a precise "specification"
  - says exactly what input/output behavior is expected
- Also want our designs to lead to code that is
  - easy to **understand**
  - easy to **test**
  - easy to **change**
- Will look at ways to achieve this

After Coding

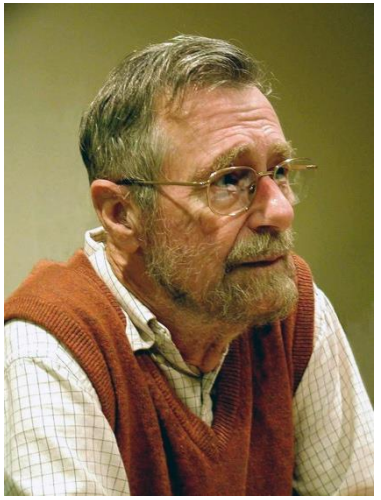
- After coding, you must ensure it is correct
  - correctness is the most important requirement
- Correctness is the hard part…
- Google search team was called "search quality"
  - hard part is measuring how good the results are
  - once you can do that, you can try various improvements

After Coding

- After coding, you must ensure it is correct
  - correctness is the most important requirement
- Correctness is the hard part…
- Once you know how to check if code is correct, you can try different ideas for how to solve it
  - see CSE 421 for broad approaches

After Coding

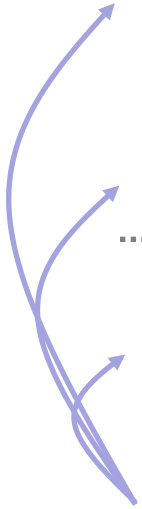
- After coding, you must ensure it is correct
- How do you do that?
- Trying on a few examples is insufficient
  - if this is enough, then AI can do that too
  - Dijkstra did not just try his algorithm on a few graphs



After Coding

- After coding, you must ensure it is correct
- How do you do that?
- CS standard is a proof of correctness
  - only way to know the code is *fully* correct for *all* inputs
  - we will focus on how to do this

Software Development Process



1. **Specification** *what*

2. Plan Generation *how*

Design

3. Coding

4. **Correctness**

Implementation

- we will not talk much about plan generation (see CSE 421)
- we will not talk much about coding (AI does that)

Course Structure

Topics

- Content organized into "topics"
  - each is one week of material
- Topics will alternate between
  - design (specifications)
  - implementation (correctness)
- 5 weeks of each, alternating between them

Lectures

- In person
  - Monday, Wednesday, and Friday in **CSE2 G01**
  - Will also be recorded

- 3 lectures on each "topic"
  - hence, one week of material
  - slides for one topic (once released) will be one PDF

Concept Checks

- One after each lecture
  - due the next day by 6pm
- Intended to keep you up-to-date with lectures
- Have as many attempts as needed
  - no reason not to get 100%
- Only need to complete 90% of them for full credit

Homework and Section

- 8 homework assignments
  - Weekly except for midterm week
- Released with section meeting on Thursdays
- Section consists of practice problems
  - Participation is graded

Exams

- Midterm in class Friday, February 13

- Final exam at an **unusual** time and place
  - on Tuesday, March 17th at 12:30
  - BAG 131/154

Course Mechanics

Websites

- All public materials posted on the website
  - lecture slides
  - homework problems
- All work submitted in Gradescope
- Everything on Canvas is private
  - do not share

Course Policies

Late Policy

- Plan to attend lecture and section
  - students who attend generally outperform those who don't

- Late policies primarily for students who are sick
  - **lectures**: will be recorded
  - **concept checks**: 90% completed will round up to 100%

    regular due date will be 6pm on day after the lecture
  - **homework**: allow 3 late days during the quarter

    at most one per assignment without special permission

Collaboration (At Home)

- Work you turn in must be your own
- Fine to work with others to figure out the problem, but not to write up your solution
  - do not take any copies of a joint work
  - wait 30 minutes before writing yours up
- AI only to be used on problems where that is indicated
  - each design HW has a problem where AI is used

Grading

Grading

- Grade average may be lower than before

- Grades are a lot less important than before
  - companies care about interviews
  - grad schools care more about recommendations
  - good chance no one else ever sees them

Grading

- **Overall scores computed as**

5%	Concept Checks
10%	Section Participation
35%	Homework
20%	Midterm Exam
30%	Final Exam

Grading

- Final grade formula TBD
  - will need to see how things look
- Broadly speaking:
  - 90% on homework and
  - 80% on final
  - means 3.5 grade

Extra Credit

- One problem in each homework
- Used to round up course grades close to cut-off
  - only if substantial effort
- If you ask me to bump your grade at the end of the quarter, I will ask you about the extra credit.

Pre-Class Material

- University standard is 2 hours of outside work for each hour in class

- Pre-class material (reading) for first week:
  - **Wed**: notes on software design
  - **Fri**: notes on mathematical notation
  - **Mon**: notes on software implementation

- Only occasional pre-class material reading for later weeks