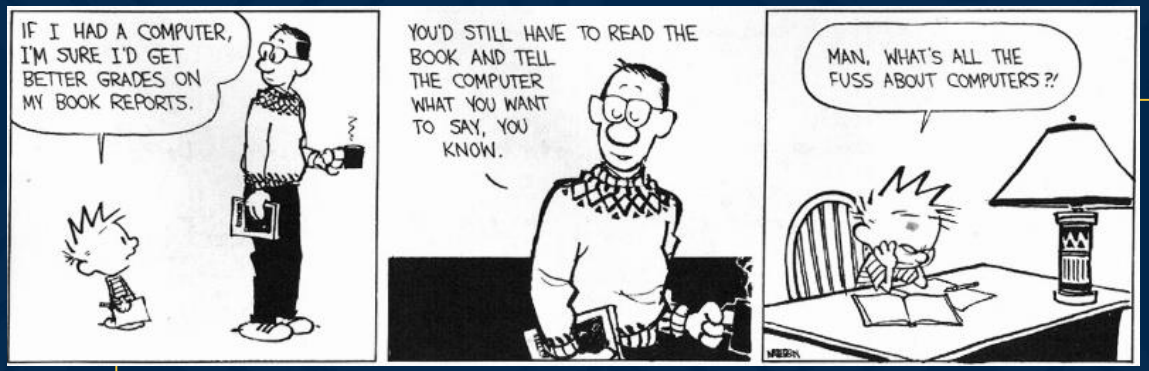
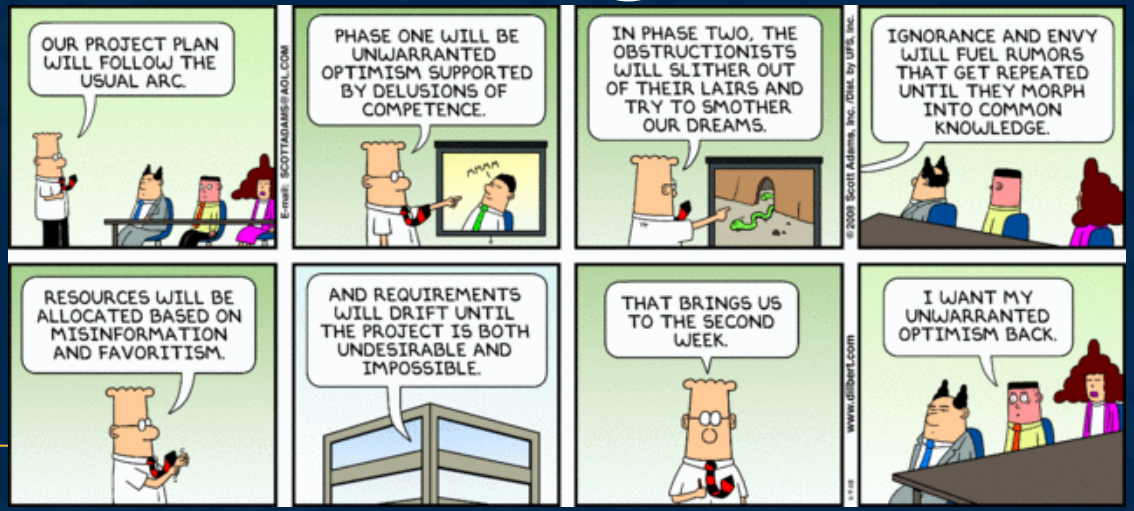


Software Engineering



Learning Objectives: by the end of today's lecture you should be able to…

1. (*Knowledge*) describe the primary activities in software engineering

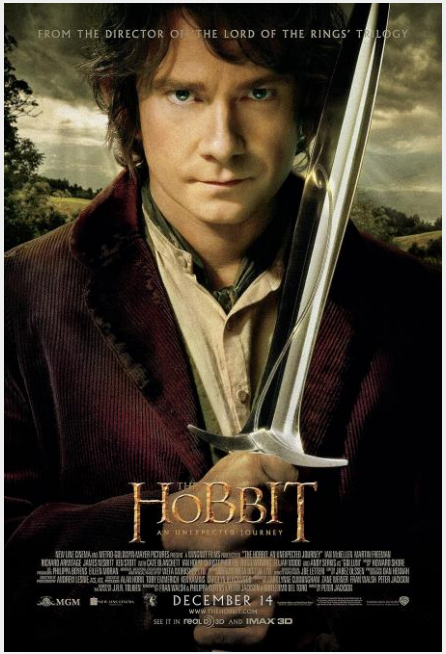
2. (*Value*) understand why software engineering is important

3. ~~(*Skill*) catch poorly thrown candy~~

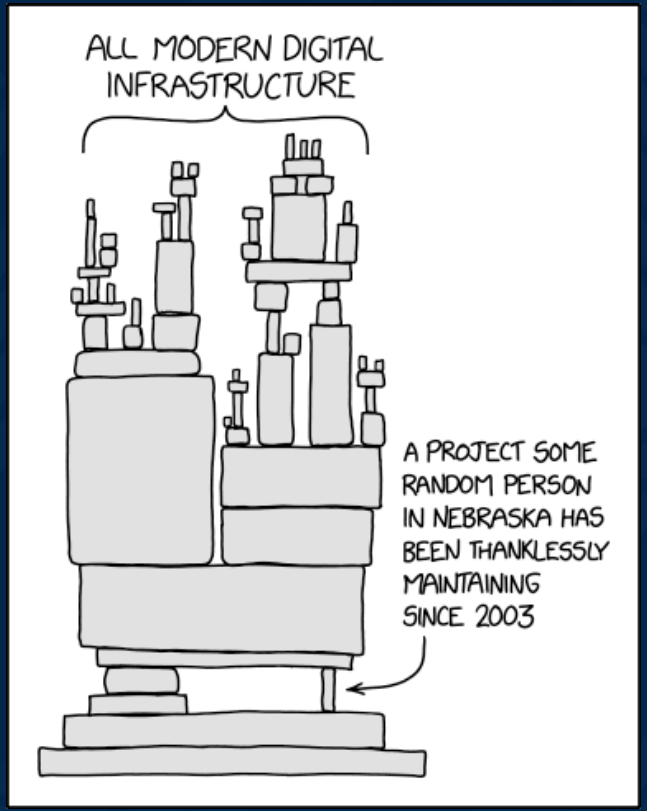
Outline

- Motivation
- What is Software Engineering?
- About this course

Our Emotional Journey…



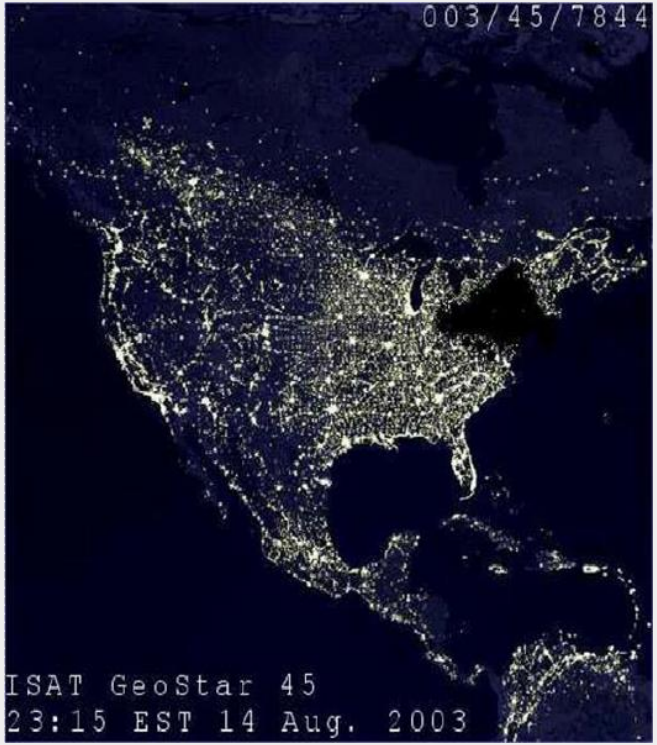
Motivation


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

List of public corporations by market capitalization

Rank	First quarter	Second quarter
1	Apple ▲2,609,000[28]	Apple ▲3,050,000[28]
2	Microsoft ▲2,146,000[29]	Microsoft ▲2,532,000[29]
3	Alphabet ▲1,332,000[30]	Alphabet ▲1,530,000[30]
4	Amazon ▲1,058,000[31]	Amazon ▲1,337,000[31]
5	Nvidia ▲686,090[32]	Nvidia ▲1,044,000[32]
6	Berkshire Hathaway ▼677,770[33]	Tesla ▲829,670[34]
7	Tesla ▲656,420[34]	Berkshire Hathaway ▲745,010[33]
8	Meta ▲549,480[36]	Meta ▲735,450[36]
9	TSMC ▲482,410[37]	TSMC ▲523,410[37]
10	Visa ▲473,870[39]	Visa ▲497,370[39]

Find The “Mitten” of Michigan

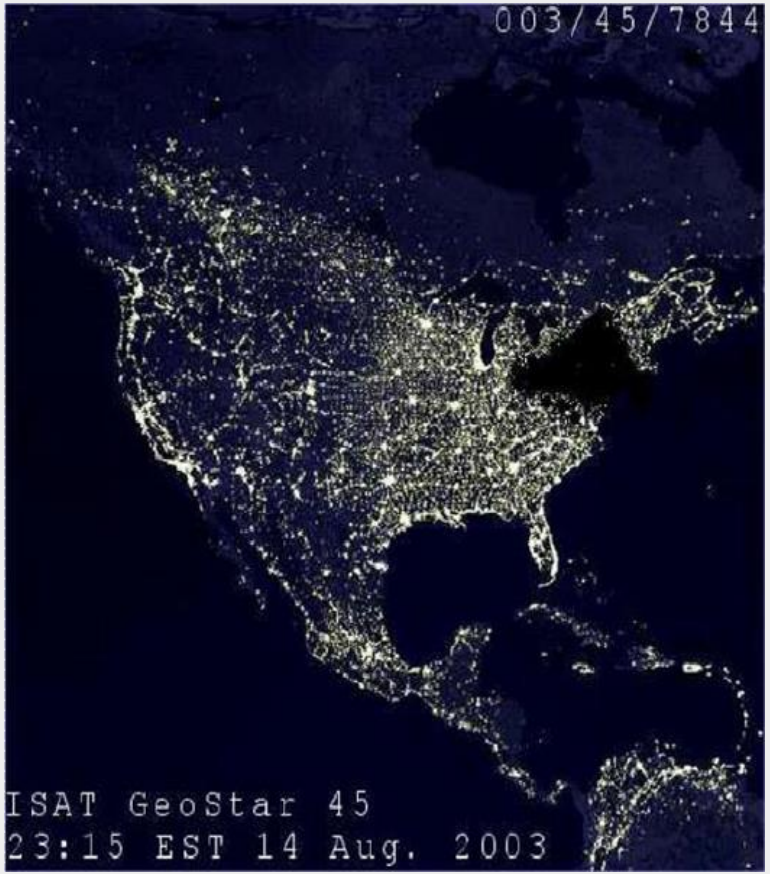


Software is Critical: Power

The **Northeast blackout of 2003** was a widespread power outage that occurred throughout parts of the Northeastern and Midwestern United States and the Canadian province of Ontario on Thursday, August 14, 2003, just after 4:10 p.m. EDT.[1]

Some power was restored by 11 p.m. Most did not get their power back until two days later. In other areas it took nearly a week or two for power to be restored.[2] At the time, it was the world's second most widespread blackout in history, after the 1999 Southern Brazil blackout.[3][4] The outage, which was much more widespread than the Northeast Blackout of 1965, affected an estimated 10 million people in Ontario and 45 million people in eight U.S. states.

The blackout's primary cause was a programming error or "bug" in the alarm system at the control room of FirstEnergy Corporation, an Akron, Ohio-based company. The lack of an alarm left operators unaware of the need to re-distribute power after overloaded transmission lines hit unpruned foliage, triggering a "race condition" in the energy management system software, a bug affecting the order of operations in the system. What would have been a manageable local blackout cascaded into massive widespread distress on the electric grid.



Software is Critical: Defense

- Quoting an Air Force lieutenant general,

  "*The only thing you can do with an F- 22 that does not require software is take a picture of it.*"

[Crouching Dragon, Hidden Software: Software in DOD Weapon Systems (Ferguson, IEEE Software, 2001)]

Software is Critical: Transportation

Computers become increasingly capable and powerful by the year and new hardware is often the most visible cue for technological progress. However, even with the shiniest hardware, the software that plays a critical role inside many systems is too often antiquated, and in some cases decades old.

This failing appears to be a key factor in why Southwest Airlines couldn't return to business as usual the way other airlines did after last week's major winter storm. More than 15,000 of its flights were canceled starting on Dec. 22, including more than 2,300 canceled this past Thursday — almost a week after the storm had passed.

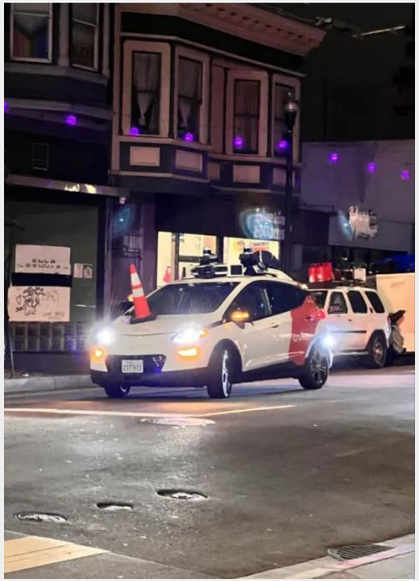
This problem — relying on older or deficient software that needs updating — is known as incurring "technical debt," meaning there is a gap between what the software needs to be and what it is. While aging code is a common cause of technical debt in older companies — such as with airlines which started automating early — it can also be found in newer systems, because software can be written in a rapid and shoddy way, rather than in a more resilient manner that makes it more dependable and easier to fix or expand. As you might expect, the former is cheaper and quicker.

The Shameful Open Secret Behind Southwest's Failure

Dec. 31, 2022

Photo Illustration by Leslie dela Vega/The New York Times; Photo by Weerayut Ranmai, via EyeEm, via

Software is Critical: Driving



Cruise cars are seen immobilized after cones were placed on their hoods by a local activist group. | Source:| Courtesy Safe Street Rebel

Carnegie Mellon

Toyota Case: Single Bit Flip That Killed

Junko Yoshida

10/25/2013 03:35 PM EDT

During the trial, embedded systems experts who reviewed Toyota's electronic throttle source code testified that they found Toyota's source code defective, and that it contains bugs -- including bugs that can cause unintended acceleration.

"We did a few things that NASA apparently did not have time to do," Barr said. For one thing, by looking within the real-time operating system, the experts identified "unprotected critical variables." They obtained and reviewed the source code for the "sub-CPU," and they "uncovered gaps and defects in the throttle fail safes."

The experts demonstrated that "the defects we found were linked to unintended acceleration through vehicle testing," Barr said. "We also obtained and reviewed the source code for the black box and found that it can record false information about the driver's actions in the final seconds before a crash."

Stack overflow and software bugs led to memory corruption, he said. And it turns out that the crux of the issue was these memory corruptions, which acted "like ricocheting bullets."

Barr also said more than half the dozens of tasks' deaths studied by the experts in their experiments "were not detected by any fail safe."

Bookout Trial Reporting

http://www.eetimes.com/document.asp?doc_id=1319903&page_number=1 (excerpts)

"Task X death in combination with other task deaths"

14

Software is Critical: Privacy

- Equifax security breach impacts 145.5 million people
  - Name, SSN, DOB, Address. Also DL# and CC#.

“I didn't have to do anything fancy,” the researcher told Motherboard, explaining that the site was vulnerable to a basic “forced browsing” bug. The researcher requested anonymity out of professional concerns. “**All you had to do was put in a search term** and get millions of results, just instantly—in cleartext, through a web app,” they said. In total, the researcher downloaded the data of hundreds of thousands of Americans in order to show Equifax the vulnerabilities within its systems. They said they could have downloaded the data of all of Equifax's customers in 10 minutes: “I've seen a lot of bad things, but not this bad.”

Software is Critical: Healthcare

Healthcare.gov: Government IT Project Failure at its Finest

Posted: 10/18/2013 6:33 pm

Read more > Project Management, Government, Healthcare, It Projects, Open Source, Business News

The *BusinessWeek* article on the Healthcare.gov failure is nothing if not instructive. From the piece:

> Healthcare.gov isn't just a website; it's more like a platform for building health-care marketplaces. Visiting the site is like visiting a restaurant. You sit in the dining room, read the menu, and tell the waiter what you want, and off he goes to the kitchen with your order. The dining room is the front end, with all the buttons to click and forms to fill out. The kitchen is the back end, with all the databases and services. The contractor most responsible for the back end is CGI Federal. Apparently it's this company's part of the system that's burning up under the load of thousands of simultaneous users.

The restaurant analogy is a good one. Projects with scopes like these fail for all sorts of reasons. *Why New Systems Fail* details a bunch of culprits, most of which are people-related.

Software is Critical:

Taylor Swift

ticketmaster

Website crash [edit]

The Eras Tour recorded an incredibly high demand for tickets.[31] On November 15, the day of the pre-sale, Ticketmaster's website crashed following "historically unprecedented demand with millions showing up", halting the presale.[2] In less than an hour of availability, the ticketing platform's servers were "unable to answer", with users experiencing turbulence—"either completely logged out or in a queue 2,000-plus people strong that appeared frozen." Ticketmaster immediately published a statement saying they are working to fix the issues "as the site was unprepared to accommodate the sheer force of hundreds of thousands of Swift fans",[18] and subsequently reported that "hundreds of thousands of tickets" had already been sold and postponed the remainder of the presale, including the Capital One presale to November 16.[32] The public on-sale was later canceled due to "extraordinarily high demands on ticketing systems and insufficient remaining ticket inventory to meet that demand".[33] The Eras Tour sold over two million tickets on its first day of presale alone, breaking the all-time record for the most concert tickets sold by an artist in a single day.[3]

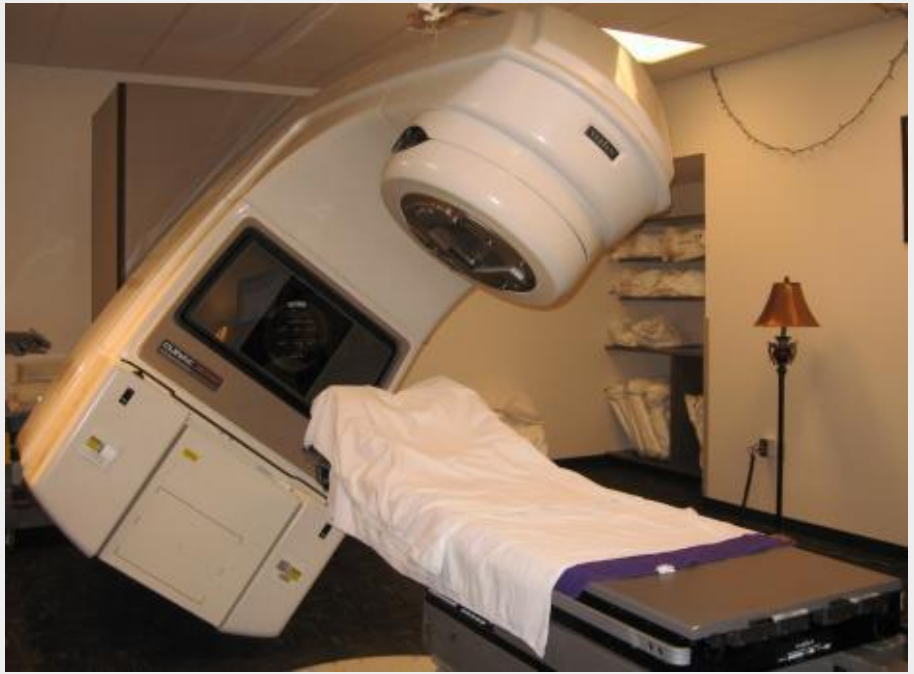
During and immediately after the website crash, Ticketmaster was widely criticized by fans and customers on social media for its ticketing model that obstructed purchasing.[34] According to *The New York Times*, the ticket sale failure "broke the internet".[4] "Ticketmaster" trended number one worldwide on various social media platforms such as Twitter and TikTok.[35] CNN Business stated that the "astronomical" demand indicated Swift's popularity.[1] However, *Fortune* and Bloomberg News attributed the criticism to Ticketmaster's "oft-confusing multistep buying process plagued with additional fees", as well as "long waits, technical problems, and poor customer service".[14][36]

Software is Critical: Space

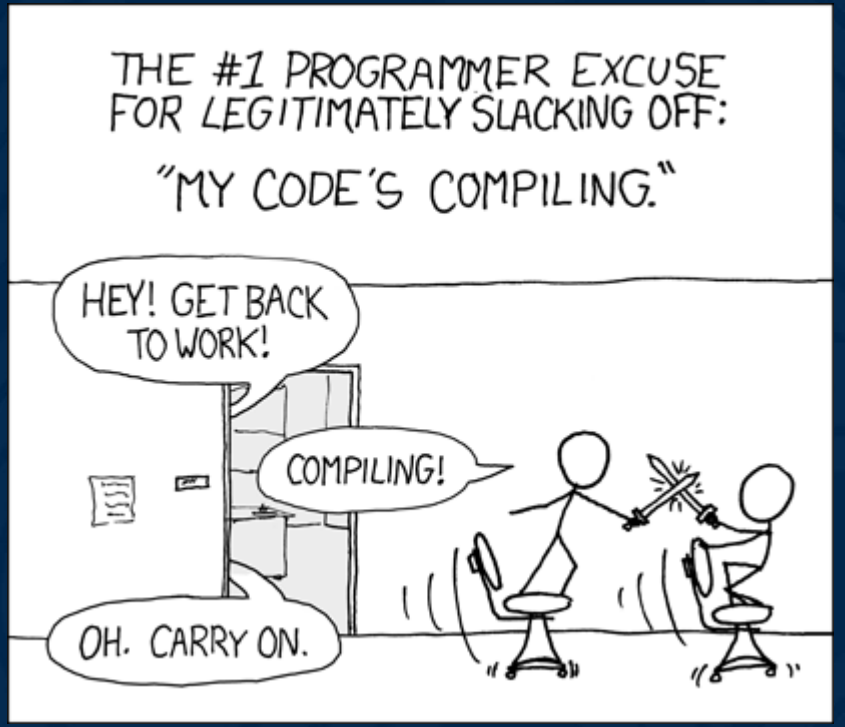
- The European Space Agency's Ariane 5 Flight 501 was destroyed 40 seconds after takeoff (June 4, 1996).
- The US$1 billion prototype rocket self-destructed due to a bug in the on-board guidance software.
- The bug? Bad conversion of `double` to `short`, leading to an overflow.

Software is Critical: Healthcare (!)

- Therac-25 radiation therapy machine
- At least six accidents in which patients were given massive overdoses of radiation
- Because of concurrent programming errors, it sometimes gave its patients radiation doses that were hundreds of times greater than normal, resulting in death or serious injury

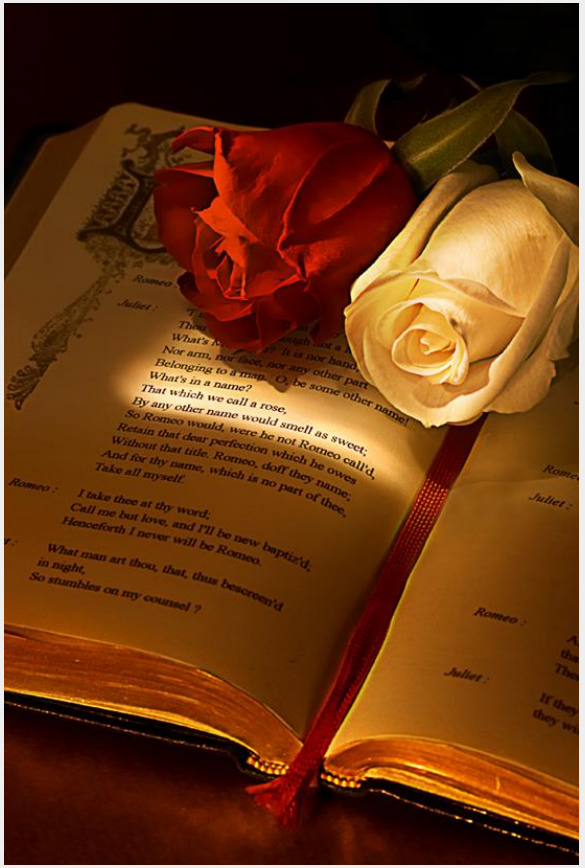


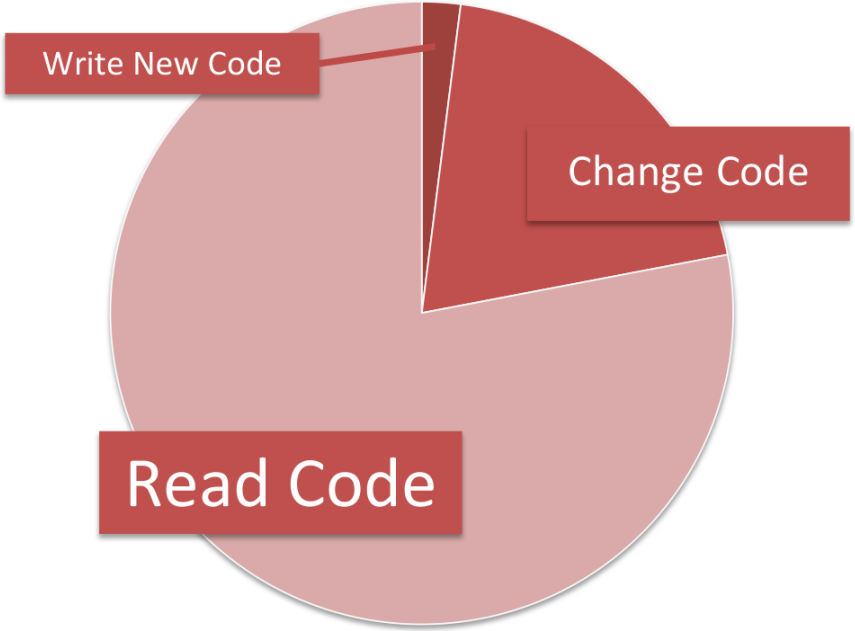
What is Software Engineering?



What is Software Engineering?

- The majority of industrial software engineering is *not* writing code.
- The dominant activities in software engineering are comprehension and maintenance.





"**Understanding code** is **by far** the activity at which professional developers spend most of their time."

[Peter Hallam. *What Do Programmers Really Do Anyway?* Microsoft.]

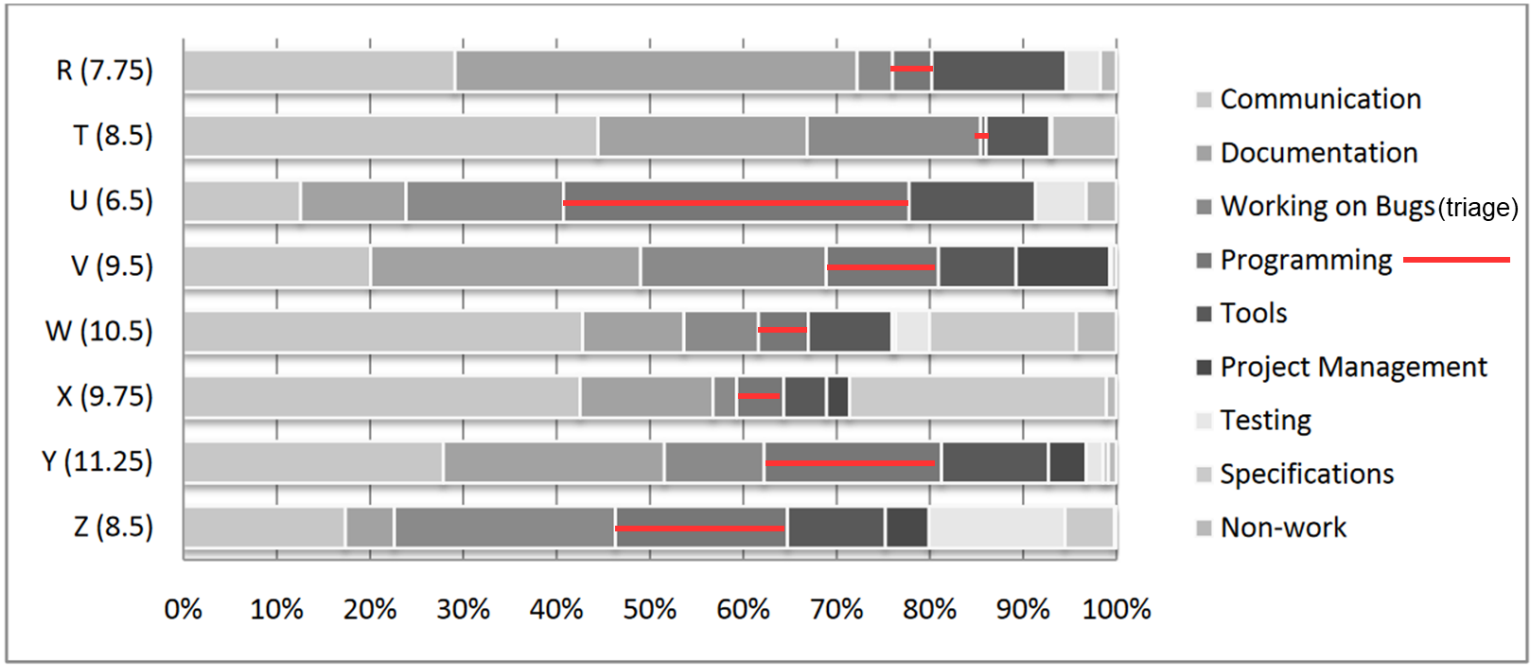
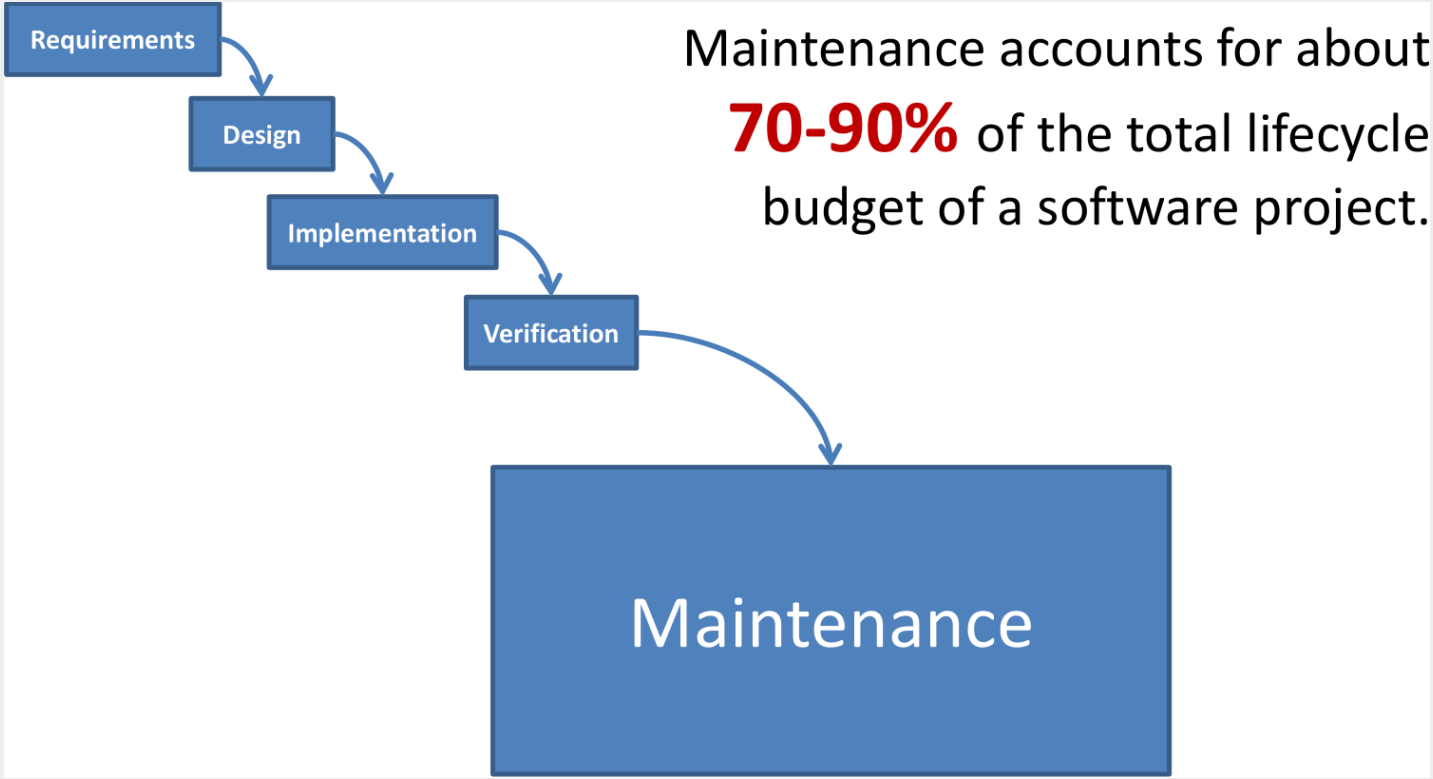


Figure 1. Tasks by time for *each* subject, normalized by the total time + time where events overlapped in each observation. Total observation time in hours is listed in parentheses after each subject's identification letter.

Hourly logging of new devs (1-7 months) at Microsoft: programming is 10-20% of the time

[Begel and Simon. *Novice Software Developers, All Over Again*. Computing Education Research, September 2008. Microsoft.]



[Pigoski. *Practical Software Maintenance: Best Practices for Managing Your Software Investment*. Seacord, Plakosh, and Lewis. *Modernizing Legacy Systems: Software Technologies*.]

IV. How Developers Spend Their Time

Figure 3 summarizes the average distribution of activities of the developers and their sessions in our dataset.

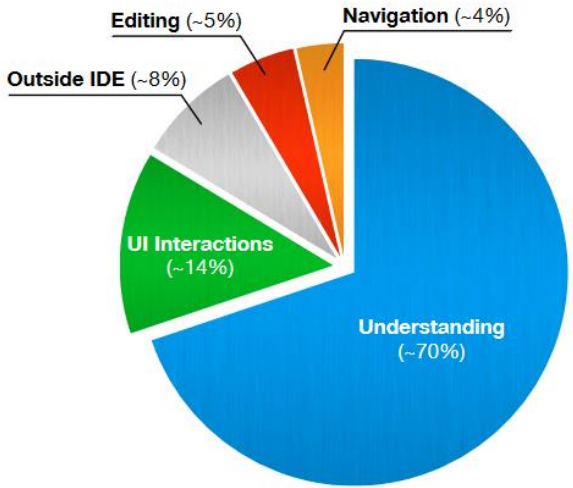
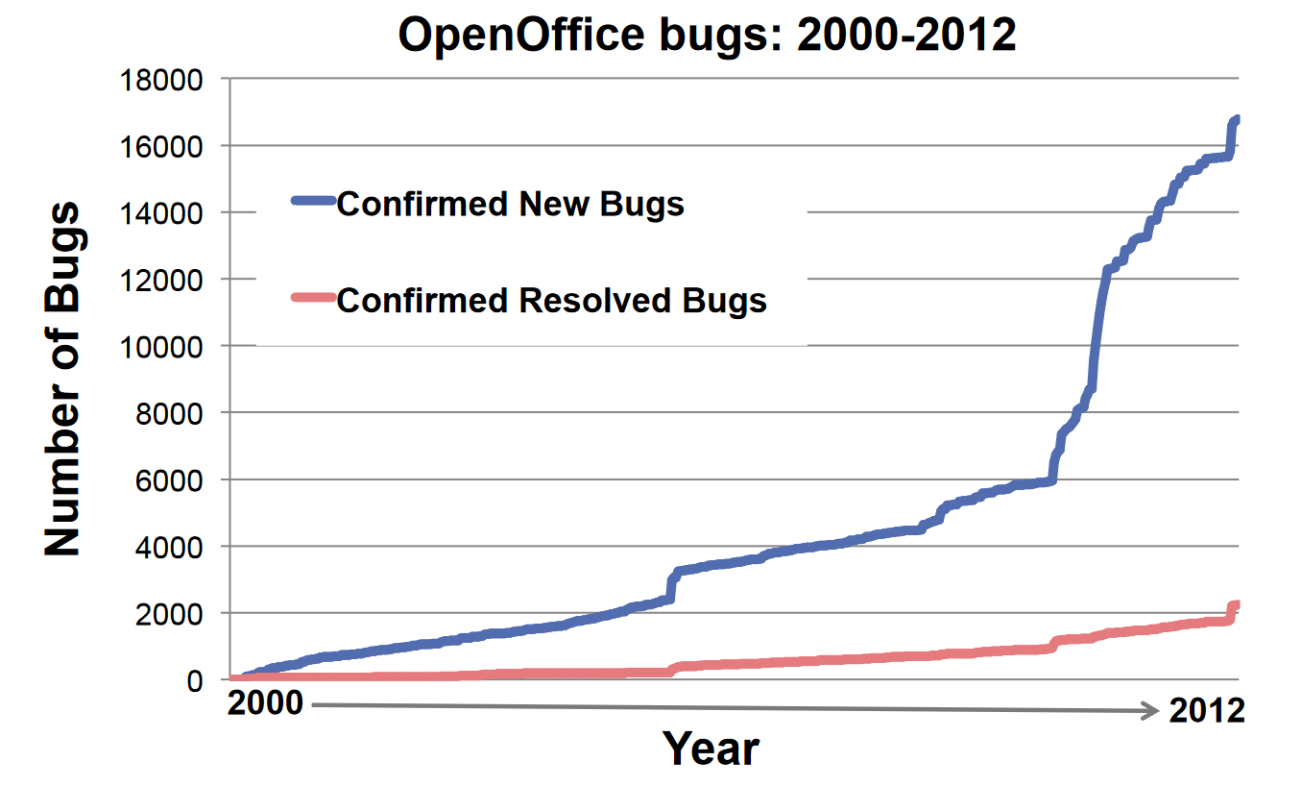


Fig. 3. How do developers spend their time?

into sprees and those into activities. In the end, 31,609 development activities originated from the 5 million events recorded with DFlow. We measured the time spent by developers in 5 distinct and disjunct categories: *understanding*, *navigation*, *editing*, *UI interactions*, and *time spent outside of the IDE*.

Our results reinforce common claims about the role of program understanding: On average, developers spend 70% of their time performing program comprehension. In addition, developers spend 14% of their time in fiddling with the UI of the IDE, which calls for novel and more efficient user interfaces. The time spent for editing and navigating source code is respectively 5% and 4%. The large part of development is occupied by mental processes (*i.e.*, understanding) and, in the remaining time, a developer has to deal with inefficient user interfaces to read, write, and browse source code. We

[Minelli, Mocci, Lanza. *I know what you did last summer: an investigation of how developers spend their time*. ICPC 2015.]

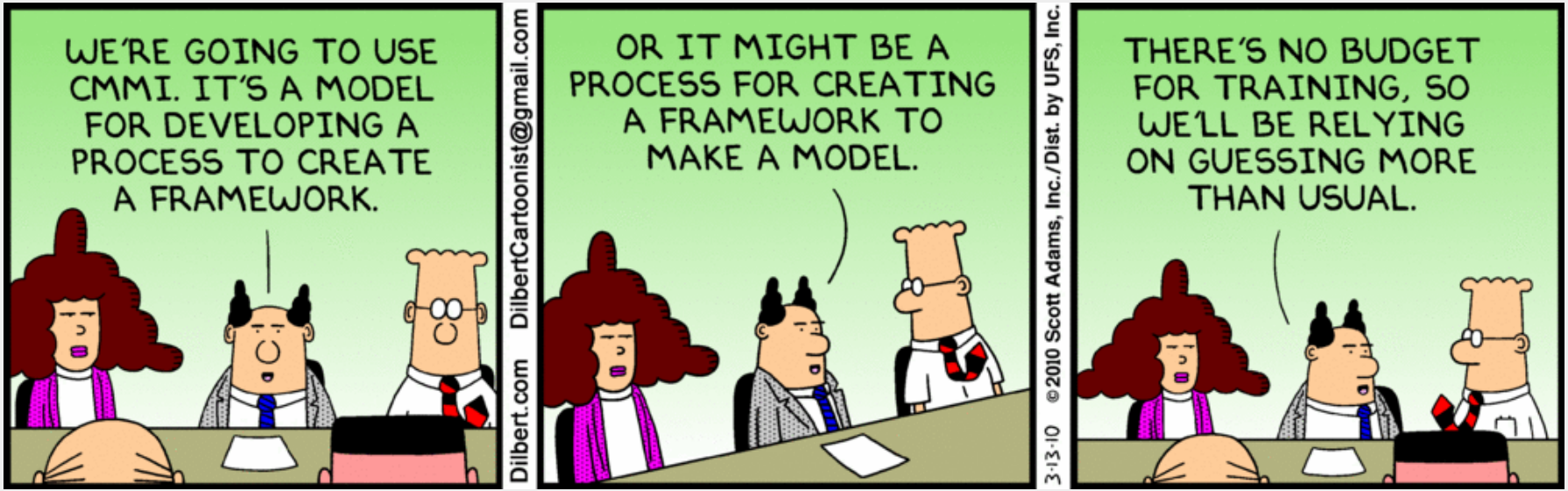


[Fry. *Leveraging Light-Weight Analyses to Aid Software Maintenance*. 2014]

About ~~Monday night~~ this course…

A Key Issue

- “Half of software engineering is crap.”
  - Wes Weimer (and your instructors)

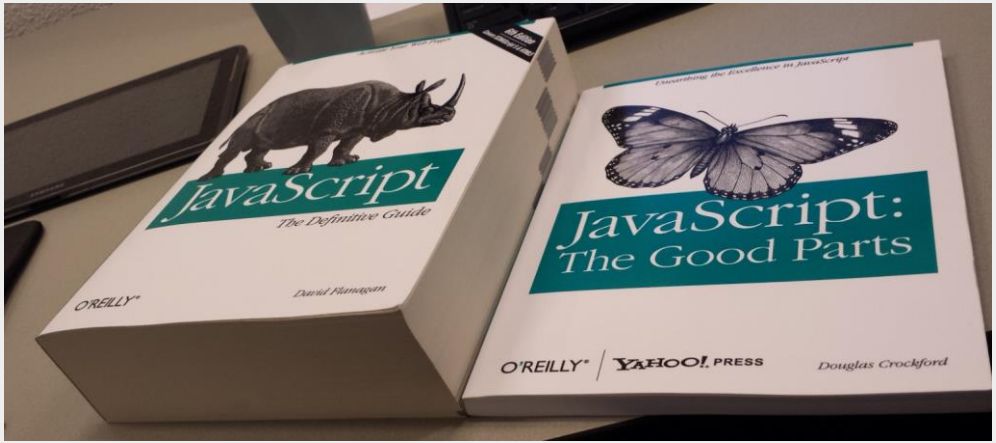


Revolutionary Solution

Class Philosophy

“Anyway, here's the 'good parts' version. S. Morgenstern wrote it. And my father read it to me. And now I give it to you. What you do with it will be of more than passing interest to us all.”

-William Goldman, *The Princess Bride*



So what about jobs

Hi Prof Weimer!

You probably don't know me but I was a student in the Winter 2020 EECS 481 class list. I just wanted to send this email to let you know how much I appreciated your class and how much knowledge from that class is already being applied to my work life today.

I started full-time work at my company about 3 weeks ago and I can confidently say nearly 100% of the material has already shown up in some form, including romance novels! (Okay maybe not that last one but everything else definitely). Practices that I've noticed at work:

- Every kind of testing (+ coverage) we've talked about has been either mentioned or put into practice
- Reading ends up being a *massive* part of my day-to-day.
- Requirements and working with customers is significantly more important that I have ever imagined... But you did warn us on that one.
- Haven't written any papers yet but it's expected of me in the future.

It's only been 3 weeks but man, I couldn't imagine everything covered in that course was being used to *this* degree. I am really glad I took your course and I hope it continues to thrive and be taught in future semesters. Although we weren't very close, I will (probably) never forget you as one of the most influential professors I've ever had!

What about recently?

Hey Wes! Hope you've been enjoying the summer.

I just started my job at Roblox, and wanted to reach out to let you know how valuable I've found 481 in just the first 3 weeks. Onboarding was (is?) intense, and I think if 481 hadn't thrown me into the deep end (with libpng's codebase and contributing to a large project), I probably would've been overwhelmed.

One of the biggest things I've noticed is that I can now keep up with senior engineers in conversations, and understand how larger infrastructure fits together.

I've actually just asked my manager for permission to investigate if we can incorporate AFL into a product I'm working on—hopefully he lets me!

Thank you so much for teaching all this. It's had a definitive positive impact on me, and I'm sure everyone else.

This Course

- https://eecs481.org/

- Administrivia
- Assignments and Grading
- Outline of Topics

We want to be here! (Brendan Kochunas)

- Have your professors worked in industry? …no
- University of Michigan (PhD in Nuclear Engineering)
  - Dissertation: "A Hybrid Parallel Algorithm for the 3-D Method of Characteristics Solution of the Boltzmann Transport Equation on High Performance Compute Clusters"
- UC Berkeley (Master's Degree in Nuclear Engineering)
- Purdue University (BS in Nuclear Engineering)
- Some internships
  - Argonne National Laboratory and Tennessee Valley Authority

Wait… why is a nuclear engineer here?

- Software from my PhD has
  - R&D 100 (2016)
  - NQA-1 Certified (2019)
  - Supported ~20 PhDs
- Also Developed graduate course “Methods and Practice of Scientific Computing” for MICDE in 2016

CASL
A DOE Energy Innovation Hub

Consortium for the Advanced Simulation of LWRs
(10 years and $250M)

What students say about Prof. Kochunas

What I wanted to tell you was THANK YOU again for your course and more importantly for talking with me after labs or lectures sometimes (as you did with so many students). You "invested" in us and you did a great, great job. I wanted to share with you one of the returns of your investment, in my case at least.

I found a postdoc opportunity posted online at NC State back in April. The project was listed as a study on the quantification of floating-point reproducibility (and other non-determinism issues) in large-scale scientific simulations on parallel systems. But this position was in the electrical and computer engineering department (ECE) and I thought that my background might be too "different" to fit in this position (coming from the civil engineering dept).

Well, I emailed the professor and applied for the position. I had some confidence because of MICDE scientific computing PhD program to put in my CV, but more importantly I had confidence because of the practical skills we learned in your course. In fact, during the interview, we talked about your course for the majority of the time because the professor saw it in my CV and basically said "tell me everything about this course, what did you learn in that class?".

She hired me after we talked about libraries, compilers, architectures, clusters, software engineering, and more... all topics we discussed and used in your class. She said, "you know, it would be so easy for me to hire a CS or ECE person for this position... but your unique combination of engineering + scientific computing is very interesting to me and I want to see what we can do together".

I am so grateful for the course you put together for us in Fall 2016: you helped shape my vocabulary and background to ace this interview and fit in to this position. You also encouraged me to apply for the IHPCSS summer school and I got in and went to Colorado for it this summer too, which has been a huge boost to my work as well.

What students say about Prof. Kochunas

Your course has been truly inspiring, allowing me to gain a profound understanding of software testing and software development routines and motivating me to continue my studies in this field.

Through the course, I have acquired invaluable skills, enabling me to apply more scientific methods and techniques to generate robust test cases, develop scalable automation for testing, and make comprehensive quality assurances.

Most importantly, they have played a crucial role in my preparation for every computer science course and my future career.

For instance, the theoretical knowledge of automation testing has empowered me to design a set of robust test cases that thoroughly assessed the functionality and performance of the slider bar I was developing. I used Python-cov to make sure that the coverage of the branches and statements is as high as possible. The knowledge and skills I have acquired from the course have profoundly enriched my academic journey, providing a robust foundation for my future endeavors in the ever-evolving field of computer science.

Moreover, aside from mastering the essential content of your course, I have also deliberately pushed myself beyond my comfort zone by collaborating with an unfamiliar individual, who later evolved into a valued friend.

Through collaborating with my partner, I vividly experienced the unparalleled advantages of pair programming, making it my favorite working style. The experience I gained from this class not only broadened my horizons but also reinforced my adaptability and interpersonal skills. My efforts were rewarded with an 'A' grade, a result that fills me with immense pride.

We want to be here! (XW)

- Have your professors worked in industry?
- Microsoft x2 (large company, worked on tools to synthesize code automatically for you, worked in RiSE team and PROSE team five years later)
- Shanghai Jiao Tong University (Electrical Engineering undergraduate)
- Also academia: best paper award 2020; Programming Language Design and Implementation, Human Factors in Computing Systems, etc.

We want to be here! (AM)

- Ali Movaghar, movaghar@umich.edu
  - B.S. in Electrical Engineering (EE), University of Tehran
  - M.S. and Ph.D. in Computer, Information and Control Engineering (CICE), The University of Michigan
  - Professor at Sharif University of Technology
  - Visiting Professor at The University of Michigan

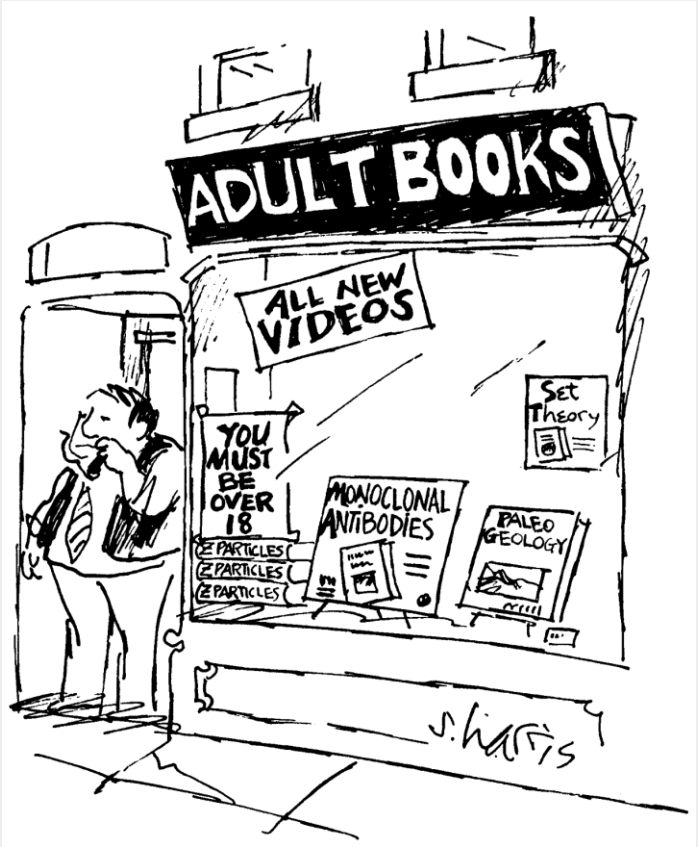
How will this help me graduate?

- Upper-level CS/CE Technical Elective
- For ENGR *and* LSA

- Major Design Experience
- Capstone

How hard is this course?

- Workload Survey is misleading!
  - Easier than 281 (Data structures) or 482 (OS)
  - Harder than 493 (UI)
- More “time consuming” than “difficult”
- See webpage quotes from former students



Assignments and Grading

- Assigned reading due before each lecture
- Normal due dates even if you add late!
- Attend lecture, take notes, visit forum
- Synchronous lecture attendance is optional
- Six homework assignments (~65%)
- Comprehension, participation (~10%)
- Two examinations (~25%)

- See webpage for regrade and makeup policy

Why participation?

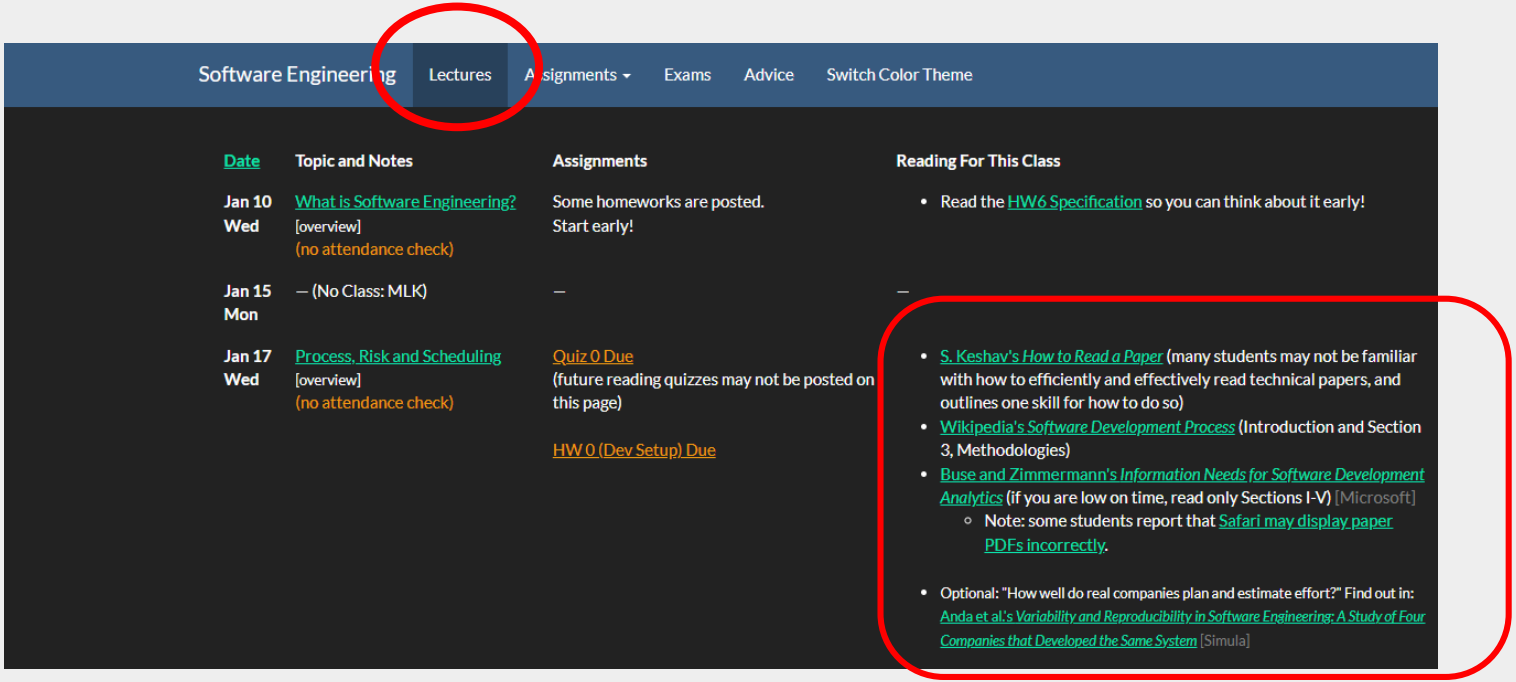
- [Kothiyal. *Effect of think-pair-share in a large CS1 class: 83% sustained engagement*. Computing Education Research 2013.]
- “This study investigated the change in critical thinking (CT) skills of baccalaureate nursing students who were educated using a Think-Pair-Share (TPS) or an equivalent Non-Think-Pair-Share (Non-TPS) teaching […] Findings revealed a significant increase in CT over time, throughout the 17-week course, with the use of TPS teaching/learning strategy.”

[Kaddoura. *Think Pair Share: A Teaching Learning Strategy to Enhance Students' Critical Thinking*. Educational Research Quarterly, 2013]

Readings

- No expensive, outdated textbook
- Assigned reading to be done before lectures
- High-level summaries (e.g. Wikipedia)
- Industrial tech reports and academic research
- Homework assignment instructions
- Optional readings for further exploration
- Higher standard than the usual EECS class

Where do I find the readings?



Assignments

- Seven Assignments
- Dev Setup, Test Coverage, Test Automation, Mutation Testing, Defect Detection, Debugging Automation, Open Source GitHub Contribution
- Coding: **`autograder.io`** (as in 280 and 490)
- Multiple object languages (C, Java, Python, etc.)
- Writing: **`gradescope`**
- Due dates posted in advance (now!)
- Materials available in advance (now!)

Optional Teams

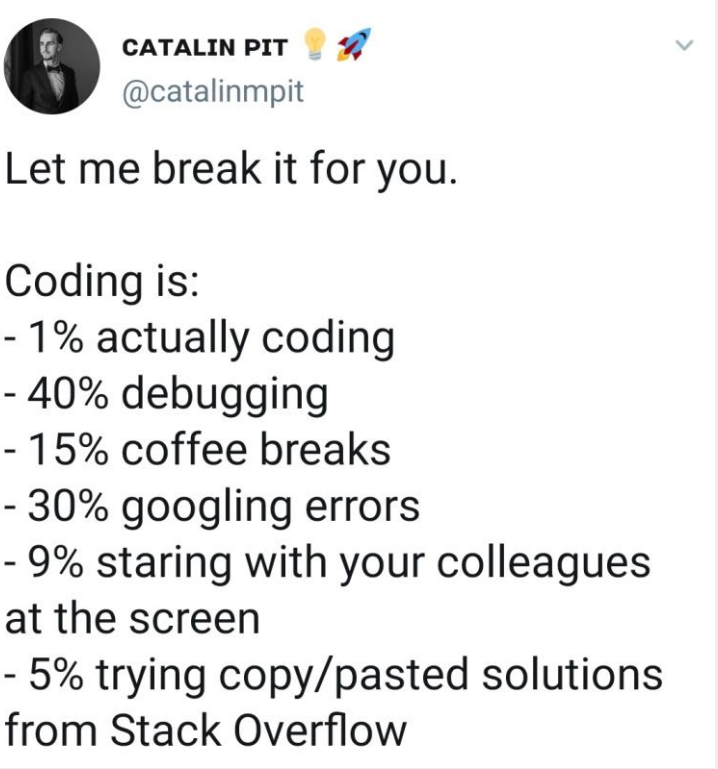
- Modern industrial software engineering is almost exclusively team-based
- But this is an ULCS, not a Capstone/MDE
- You will be exposed to building a large project in a team elsewhere in the curriculum
- For most of the assignments, you may work **alone or in pairs** of your own choosing
- We are not responsible if you partner disappears
- Use the forum to find partners, etc.

In Person vs. Remote Participation

- Live Synchronous Lecture (All Sections 3:00 in STAMPS)
  - You can choose to attend a lecture in person and receive participation credit for that lecture by completing an **in-class in-person** activity. These activities typically involve writing an answer on notecards that we pass around or completing in-class coding; they include an aspect of (sampled) "checking attendance". We collect notecards by the end of the class. We do not accept submissions after class.
- Remote Asynchronous Participation
  - In case you don't attend a lecture in person (for whatever reason), you can receive participation credit for that lecture by completing an offline "structured activity". Such activities serve as a form of participation check. It typically requires: (1) watching the lecture recording, (2) answering the same notecard question, and (3) **attempting** to solve a problem related to the lecture content.
  - The activities are a little longer, but you can do them whenever you want and have more time to do them
  - Recordings of Lecture are always available, practice activity instructions are always available: you never have to miss anything!
- Mix and Match!

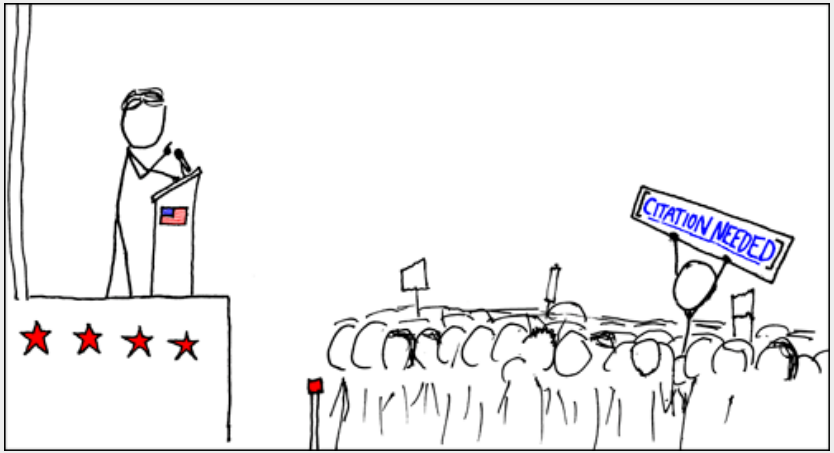
Discussion Sections

- Homework help (!), exam preparation, explain difficult material, answer questions
- **Primary Discussion Section**
  - “Lecture”-style, will be recorded for everyone to see
- Other Discussion Sections are like Office Hours
  - We know you can attend them
- Online OH will use a queue
- No required Discussion attendance
- Attend any! Or mix & match



Software Engineering You Can Believe In!

- Citations for strong claims (or ask on forum)
- Guest Lectures (we hope!)
  - Large companies, startups, etc.
- Readings from Industry
- Material from
  - Prem Devanbu
  - Christian Kästner
  - Marouane Kessentini
  - Kevin Leach
  - Claire Le Goues



Changeups and Trivia

Why Changeups and Trivia?

- “[Professors who] deliberately and consistently interspersed their lectures with … some other form of deliberate break … usually commanded a better attention span from the class, and these deliberate variations had the effect of postponing or even eliminating the occurrence of an attention break”

[Johnstone and Percival. *Attention breaks in lectures.* Education in Chemistry, 13. 49-50, 1976.]

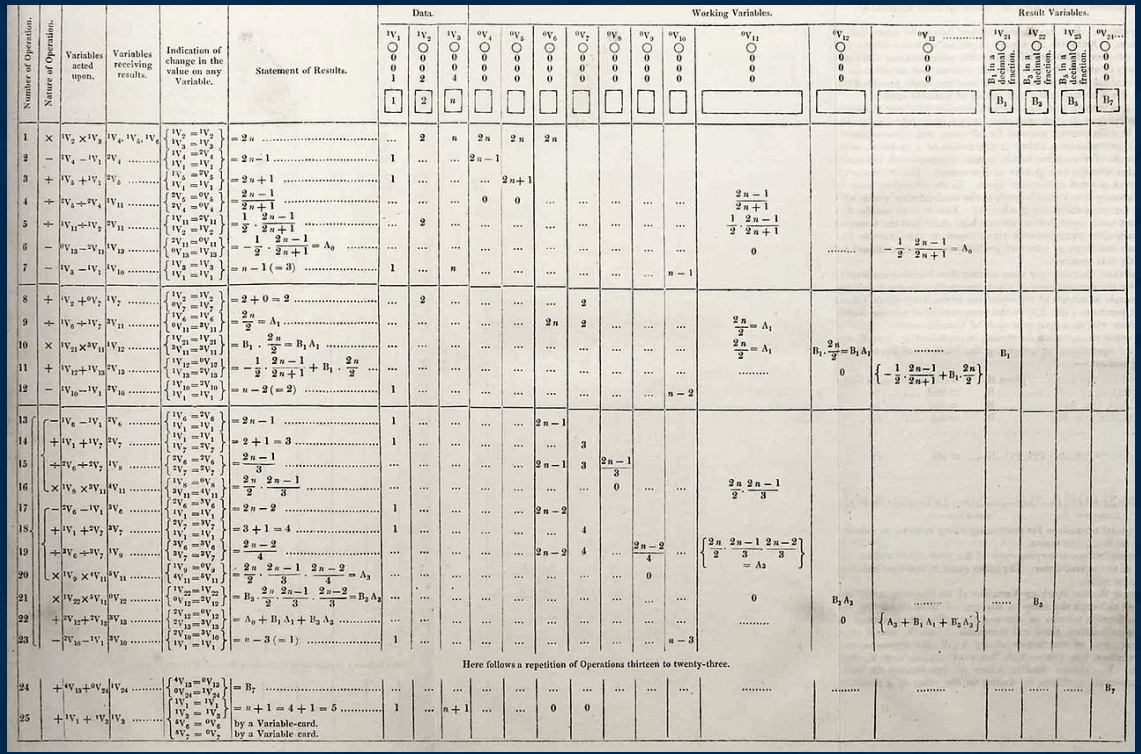
[Middendorf and Kalish. *The “Change–up” in Lectures*. TRC Newsletter, 8:1 (Fall 1996).]

Computer Science Trivia

- *This* English mathematician and writer published the first algorithm (~1842) to be carried out by a general-purpose computer and is often called the first computer programmer.

Computer Science Trivia

- *What* did that first program do?

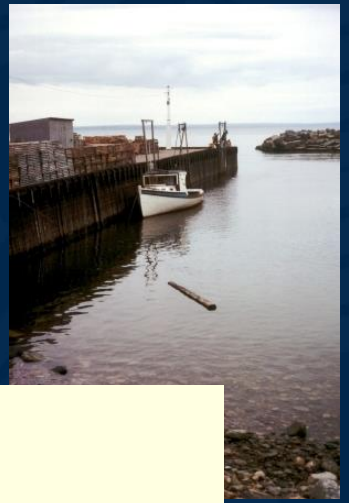


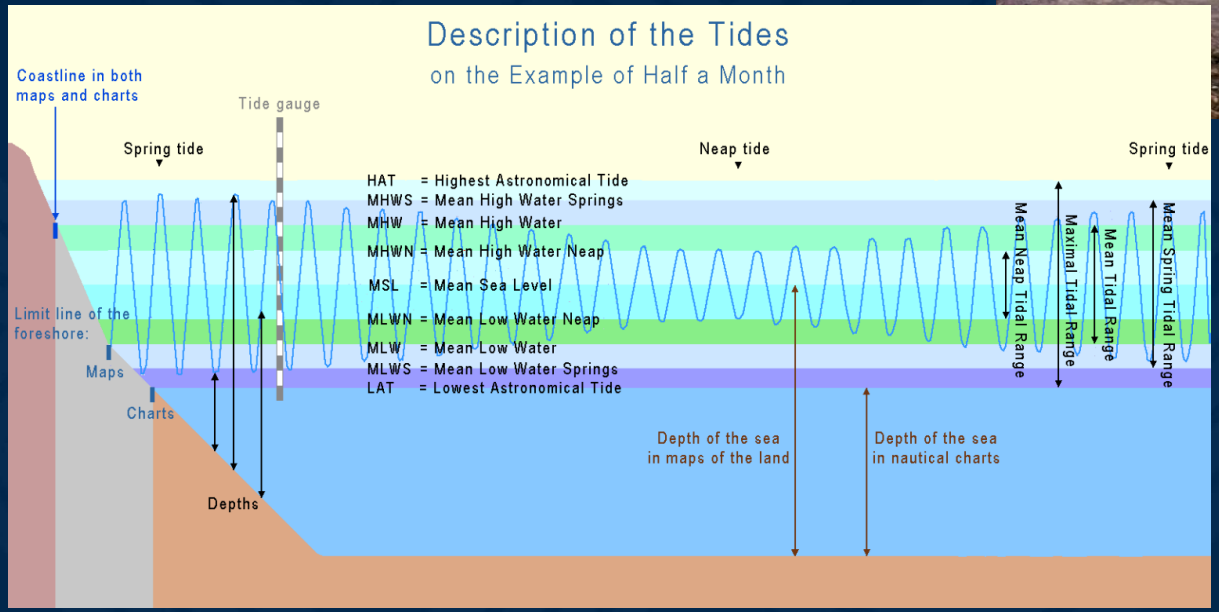
[Michael S. Reidy. Tides of History: Ocean Science and Her Majesty's Navy. From Chapter 1, “Tidal Prediction After Newton and Halley”]

Who Cared? What was one “killer app”?

A hundred years later, the early Victorians used not Newton's theory of the tides but the variant that Bernoulli worked out for this competition, which they referred to as the equilibrium theory. The equilibrium theory outlined the essential mathematical formulas needed to predict the tides for each day of the year based on the positions of the sun and moon. The most important variable for each port was the “vulgar establishment,’“ published more than ninety years later. Bernoulli's treatise significantly advanced the methods of tidal prediction, but using those methods still demanded both theoretical refinement and intense and laborious calculations. Moreover, to find the corrected establishment, one also to calculate the important variables specific to each port. Observations of the tides were a valued commodity, however, and once taken they were guarded as private property. The inaccessibility of Bernoulli's methods

“Amazon Prime” circa 1842





Psychology: The Fundamental Attribution Error

- The fundamental attribution error is that people emphasize internal characteristics when explaining the behavior of others but external factors when explaining their own behavior.
  - Example: cutting someone off in traffic.
- In an experiment, subjects read essays for and against Fidel Castro and were asked to rate the pro-Castro attributes of the writers. Conditions:
  - When subjects believed the writers chose freely:
    - Expect “pro-Castro” → positive attitude
  - When subjects believed the positions were determined by a coin toss:
    - Expect neutral attitude on average

Psychology: The Fundamental Attribution Error

- Experimental Findings:
  - Even when they knew the position came from a coin toss, subjects rated pro-Castro essay writers as having a positive Castro attitude.
  - “The subjects were unable to properly see the influence of the situational constraints placed upon the writers; they could not refrain from attributing sincere belief to the writers.”

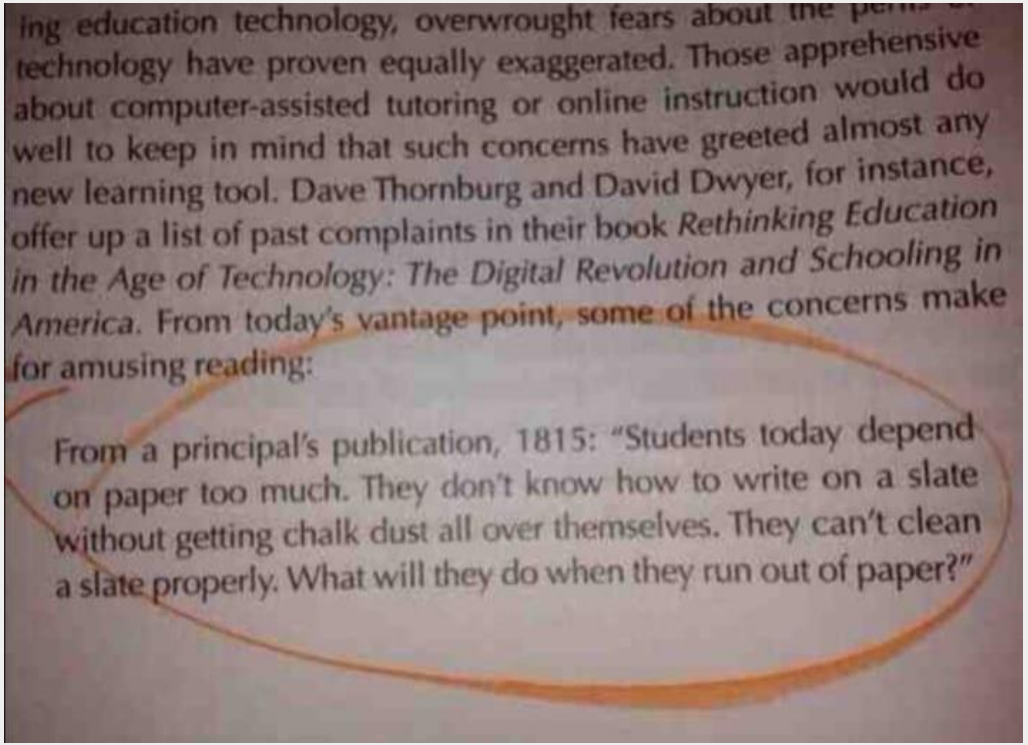
[Jones, E. E.; Harris, V. A. (1967). *"The attribution of attitudes"*. Journal of Experimental Social Psychology. 3 (1): 1–24.]

- SE Implication: Teamwork. Be careful when you see defects (mine just mean I made a typo, others mean they are stupid).

About this course (continued)

Blah blah blah laptops, blah …

(uphill both ways…)



ing education technology, overwrought fears about the pe
technology have proven equally exaggerated. Those apprehensive about computer-assisted tutoring or online instruction would do well to keep in mind that such concerns have greeted almost any new learning tool. Dave Thornburg and David Dwyer, for instance, offer up a list of past complaints in their book *Rethinking Education in the Age of Technology: The Digital Revolution and Schooling in America*. From today's vantage point, some of the concerns make for amusing reading:

From a principal's publication, 1815: "Students today depend on paper too much. They don't know how to write on a slate without getting chalk dust all over themselves. They can't clean a slate properly. What will they do when they run out of paper?"

Laptops and Cell Phones

"…participants who multitasked on a laptop during lecture scored lower on a test compared to those who did not multitask, and participants who were in direct view of a multitasking peer scored lower on a test compared to those who were not. The results demonstrate that multitasking on a laptop poses a significant distraction to both users and **fellow students** and can be detrimental to comprehension of lecture content."

[Faria Sana, Tina Weston, and Nicholas J. Cepeda. 2013. *Laptop multitasking hinders classroom learning for both users and nearby peers.* Comput. Educ. 62 (March 2013), 24-31.]

Laptops and Cell Phones

"…students who took notes on laptops performed worse on conceptual questions than students who took notes longhand. We show that whereas taking more notes can be beneficial, laptop note takers' tendency to transcribe lectures verbatim rather than processing information and reframing it in their own words is detrimental to learning."

[Mueller PA1, Oppenheimer DM2. *The pen is mightier than the keyboard: advantages of longhand over laptop note taking.* Psychol Sci. 2014 Jun; 25(6):Epub 2014 Apr 23.]

Core Course Topics

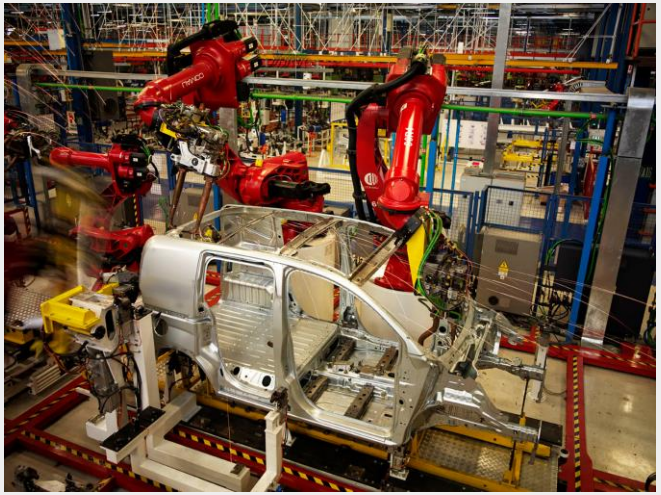
- Measurement and Risk
- Process, scheduling, and information
- **Quality Assurance**
- Code review, **testing**, and analysis
- Software Defects
- Reporting and localizing
- Software Design
- Requirements, patterns, and maintainability
- Productivity at Scale
- People, teams, interviews, synthesis, and brains

Course Themes

- Software engineering is a human process
- Software engineering deals with large scales
- Software engineering requires strategic thinking
- Software engineering is constrained by reality

Analogy: Engineering Envy

- Producing a car
  (or a nuclear power plant!)
- Estimate costs, risks
- Expected results
- High quality
- Separate plan and production
- Simulate before constructing
- Quality assurance through measurement
- Potential for automation



Dangerous Analogy

- Producing a car (or a nuclear power plant!)
- Estimate costs, risks
- Expected results
- High quality
- Separate plan and production
- Simulate before constructing
- Quality assurance through measurement
- Potential for automation

- Software = Design = Plan
- Programming is design, not production
- Production (copying/loading a program) is automated
- Simulation is not necessary
- Quality measurement?

Software Engineering

- “My favorite operational definition of engineering is ‘**design under constraint**’. Engineering is creating, designing what can be, but it is constrained by nature, by cost, by concerns of safety, reliability, environmental impact, manufacturability, maintainability, and many other such ‘ilities.’”

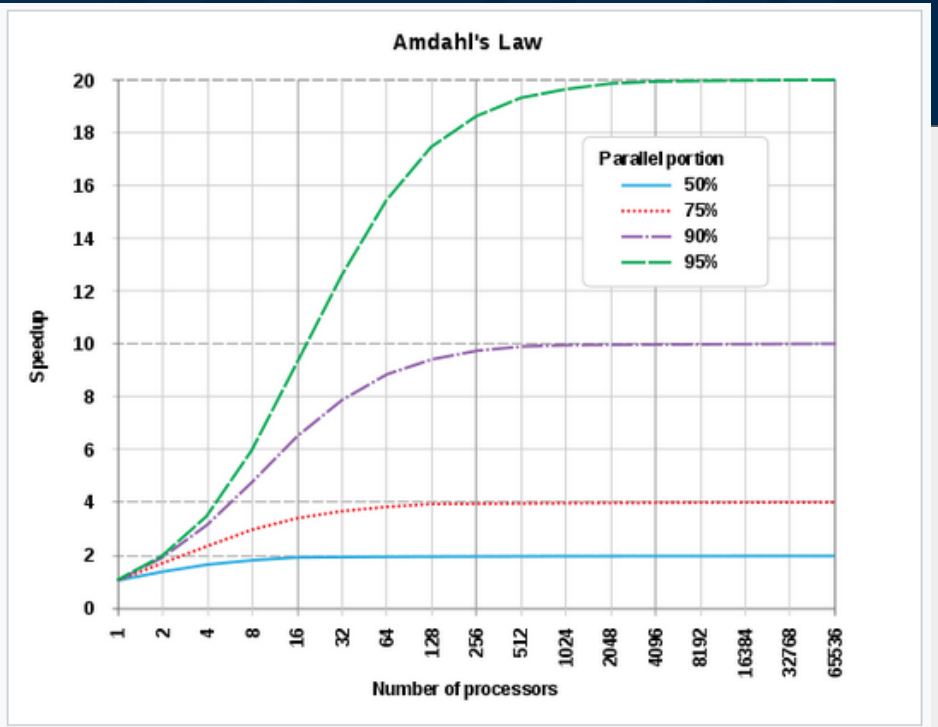
  [Bill Wulf, NAE President, The Urgency of Engineering Education Reform, 2008]

- “[Software Engineering is] The Establishment and use of sound **engineering principles** in order to obtain **economically** software that is **reliable** and works **efficiently** on **real** machines.”

  [Bauer 1975, S. 524]

Measurement Teaser

- What is Amdahl's Law?
- Suppose you want a program to run faster
- Suppose you want software to be created-and-sold faster?



Evolution according to Amdahl's law of the theoretical speedup in latency of the execution of a program in function of the number of processors executing it, for different values of p. The speedup is limited by the serial part of the program. For example, if 95% of the program can be parallelized, the theoretical maximum speedup using parallel computing would be 20 times.

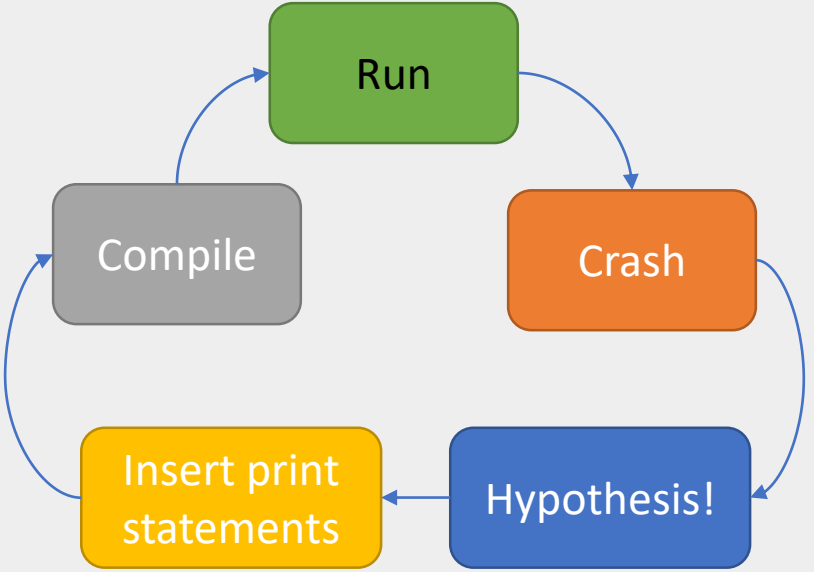
Quality Assurance Teaser

- To assess quality, we can look at the source code or run the program
- Testing is the dominant approach here
- But not all test suites are created equal!
- Statement coverage, branch coverage

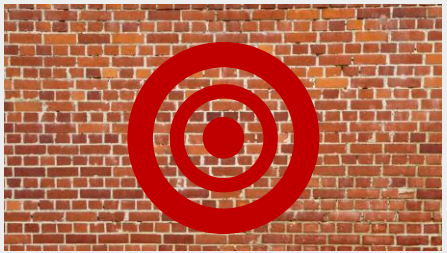
- Mutation testing
- Automated test generation

Defect Teaser

- Just put in print statements
- Find the line with the bug
- Flail around, resubmit until it passes



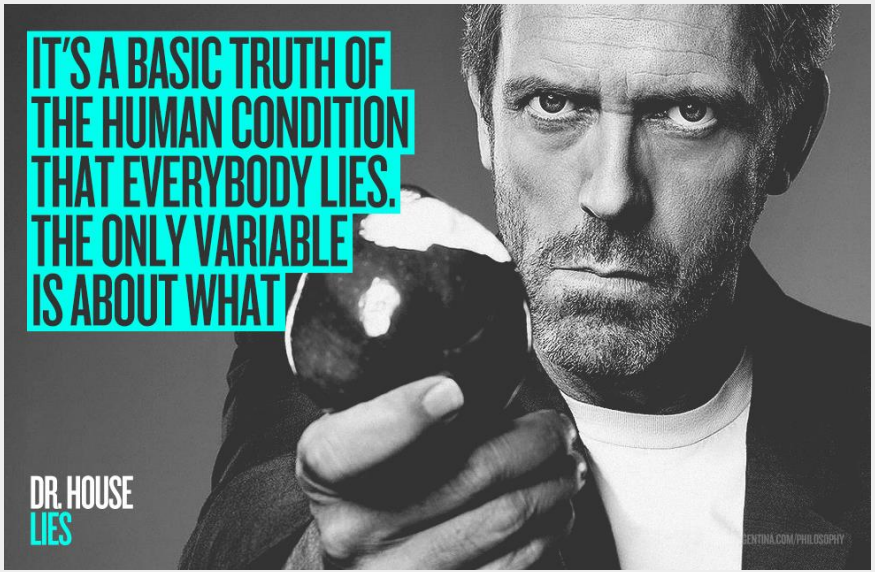
- Automatic fault localization
- Debugging as Hypothesis Testing



Bang Head Here

Design Teaser

- Requirements and Specifications
- How can we elicit what people actually want?
- Validation and Risk
- Design for Maintainability



Productivity Teaser

- The ratio of programming time and program performance between novices and experts has been published at up to 28:1

- Why?

- Pair Programming, Agile, etc.
- How do experts and novices think?
- Medical Imaging Studies

Automation and Scale

- "[some new tool] allows me to partially guide the synthesizer if I know the next few steps—I don't have to know the entire solution, but I know how to start and I can let the synthesizer fill in the holes."

- Can ~~ChatGPT~~ tools really write high-quality code for us?

- How are big companies like Facebook deploying automated bug repair or code synthesis?

Questions?

- You are responsible for all assignments at their listed times even if you add the course late.

- No office hours this week
- No discussion this week