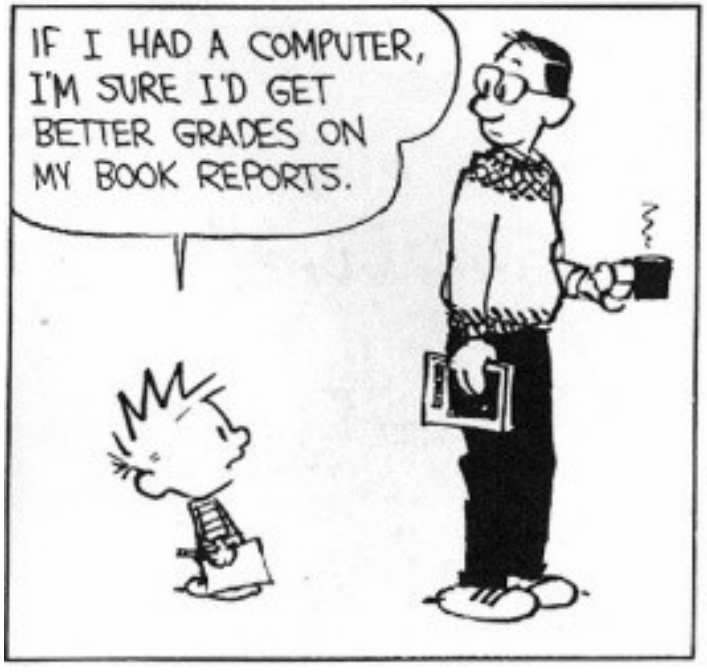
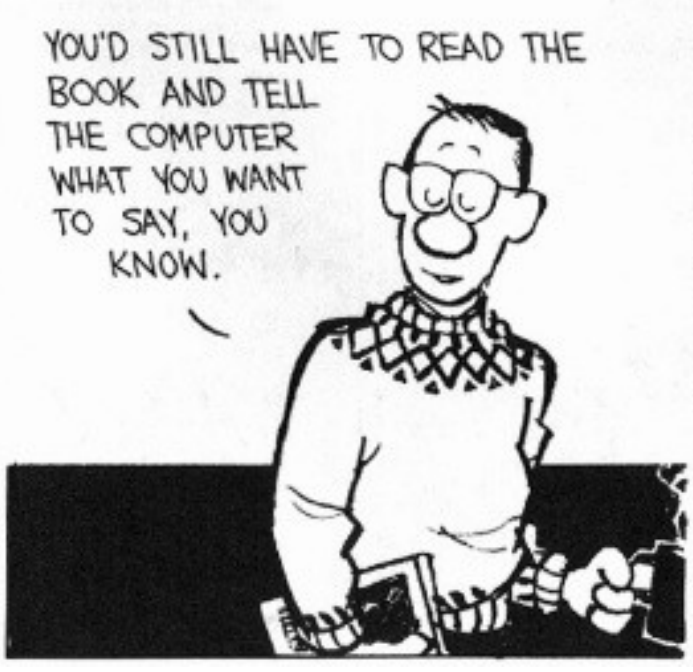
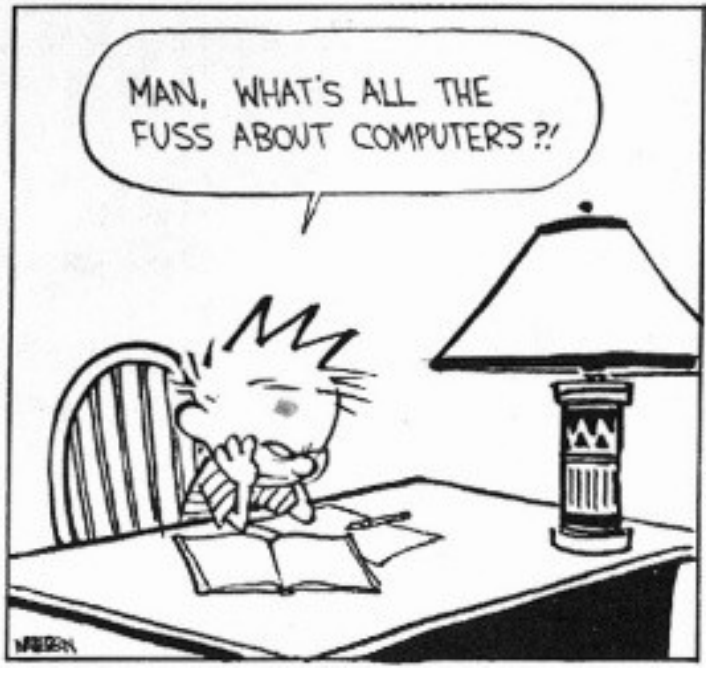
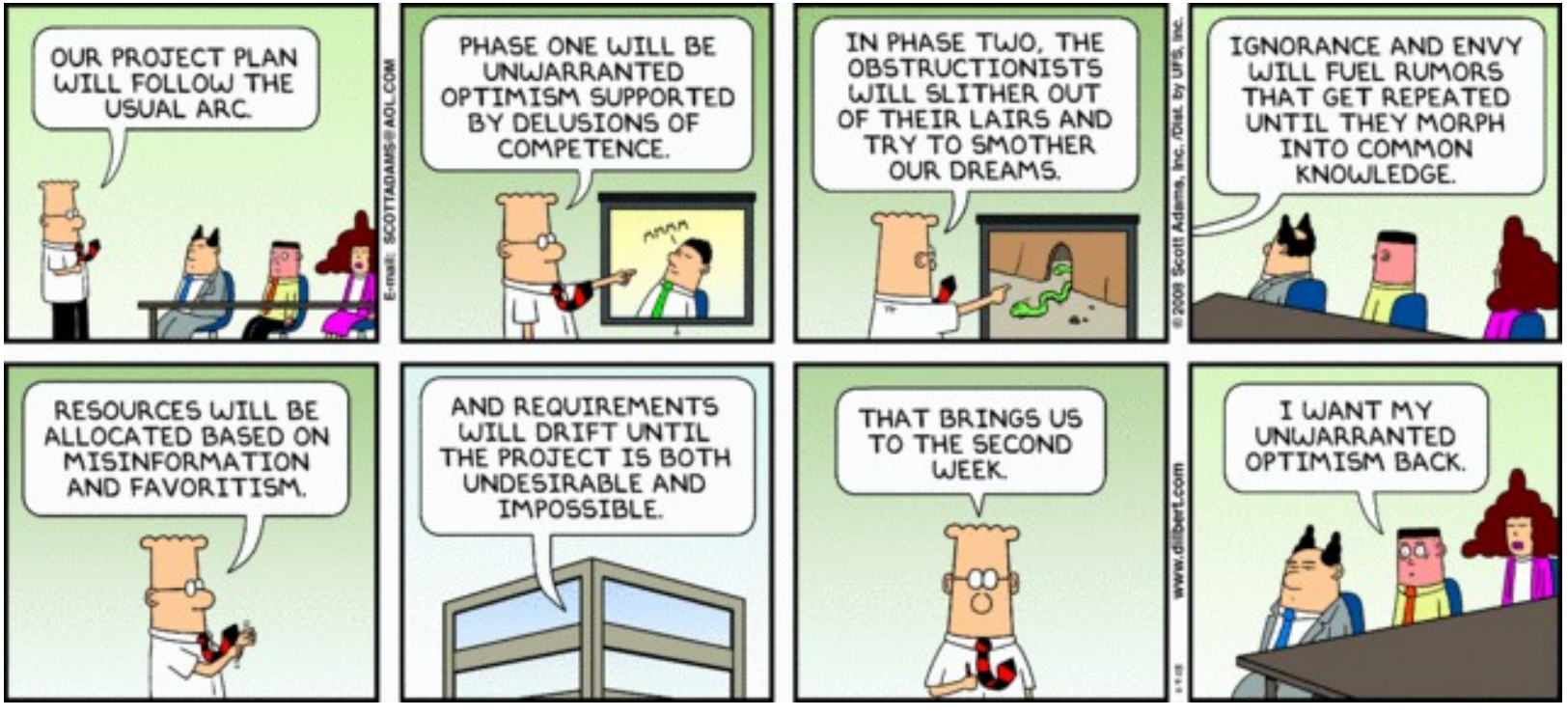






Software Engineering

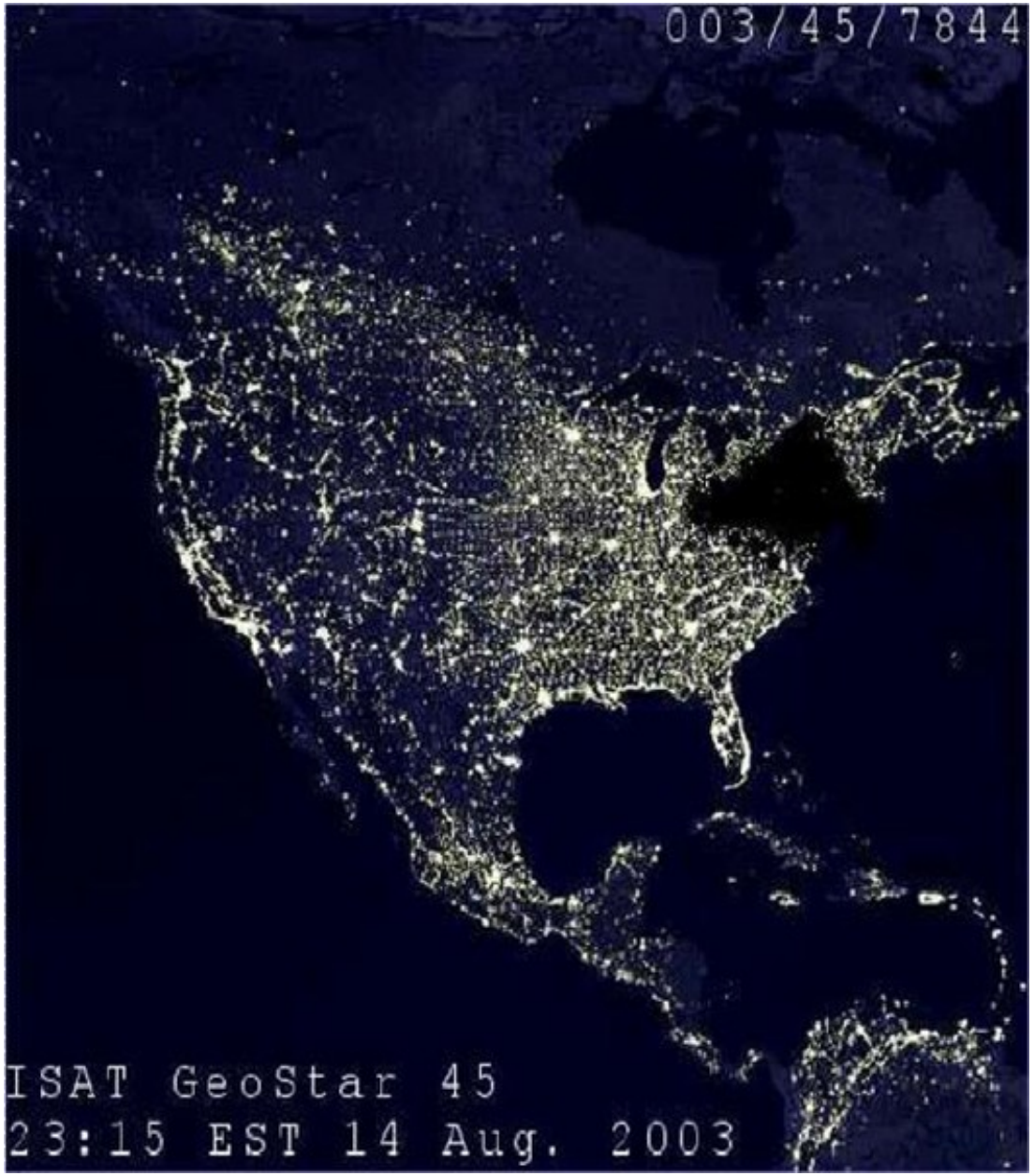


List of public corporations by market capitalization

(as of Dec 31, 2021)

Third quarter		Fourth quarter	
	Apple ▲2,339,000[19]		Apple ▲2,913,000[19]
	Microsoft ▲2,119,000[20]		Microsoft ▲2,525,000[20]
	Alphabet ▲1,777,000[22]		Alphabet ▲1,922,000[22]
	Amazon ▼1,664,000[21]		Amazon ▲1,691,000[21]
	Meta ▼956,890[23]		Tesla ▲1,061,000[24]
	Tesla ▲776,850[24]		Meta ▼935,640[23]
	Berkshire Hathaway ▼619,950[26]		Nvidia ▲732,920[27]
	TSMC ▼579,030[29]		Berkshire Hathaway ▲668,630[30]
	Tencent ▼574,460[25]		TSMC ▲623,930[29]
	Nvidia ▲517,900[27]		Tencent ▼559,900[25]

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: Driving

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"

Software is Critical: Privacy

- Equifax security breach impacts 145.5 million
  - 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.

As I read the article, a few other things jumped out at me, as they virtually guarantee failure:

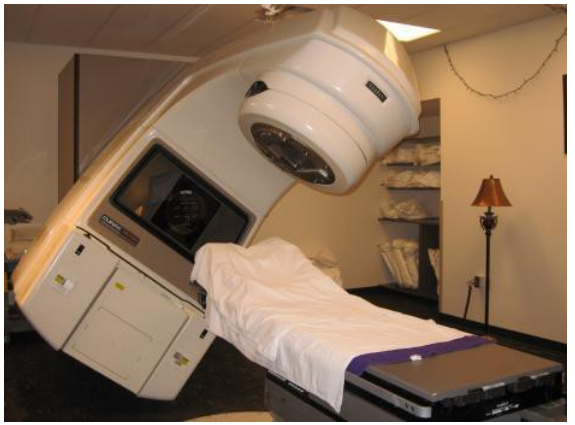
- The sheer number of vendors involved

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

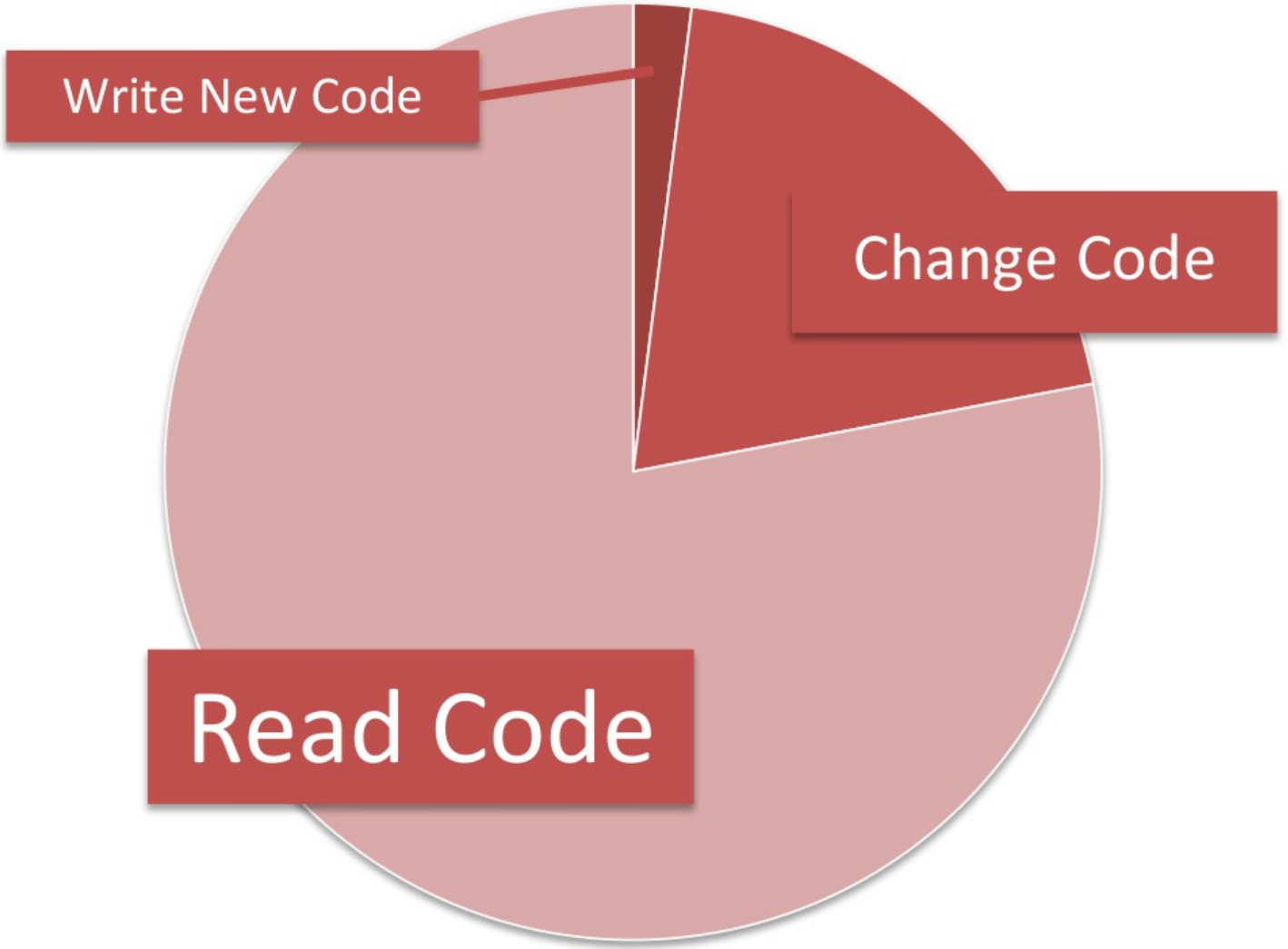


What Is Software Engineering?

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.]

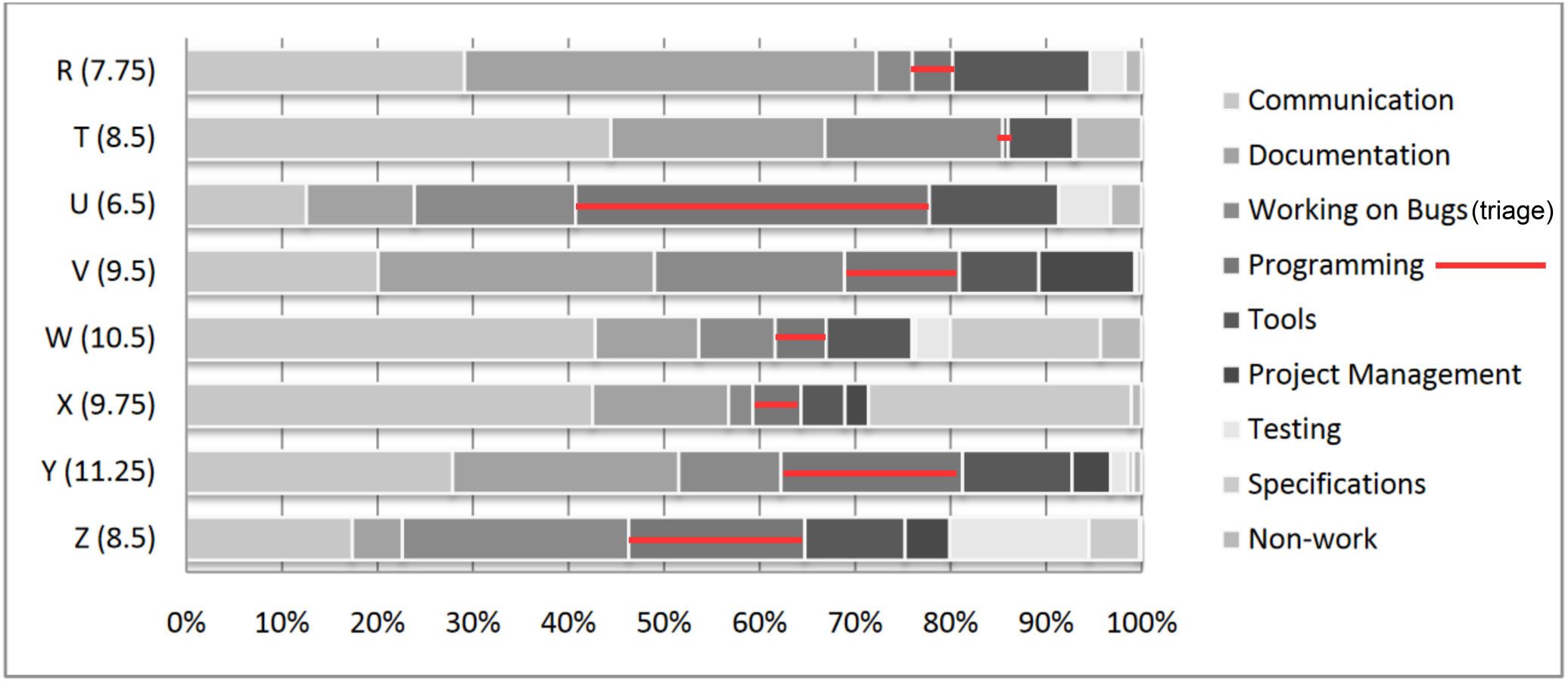
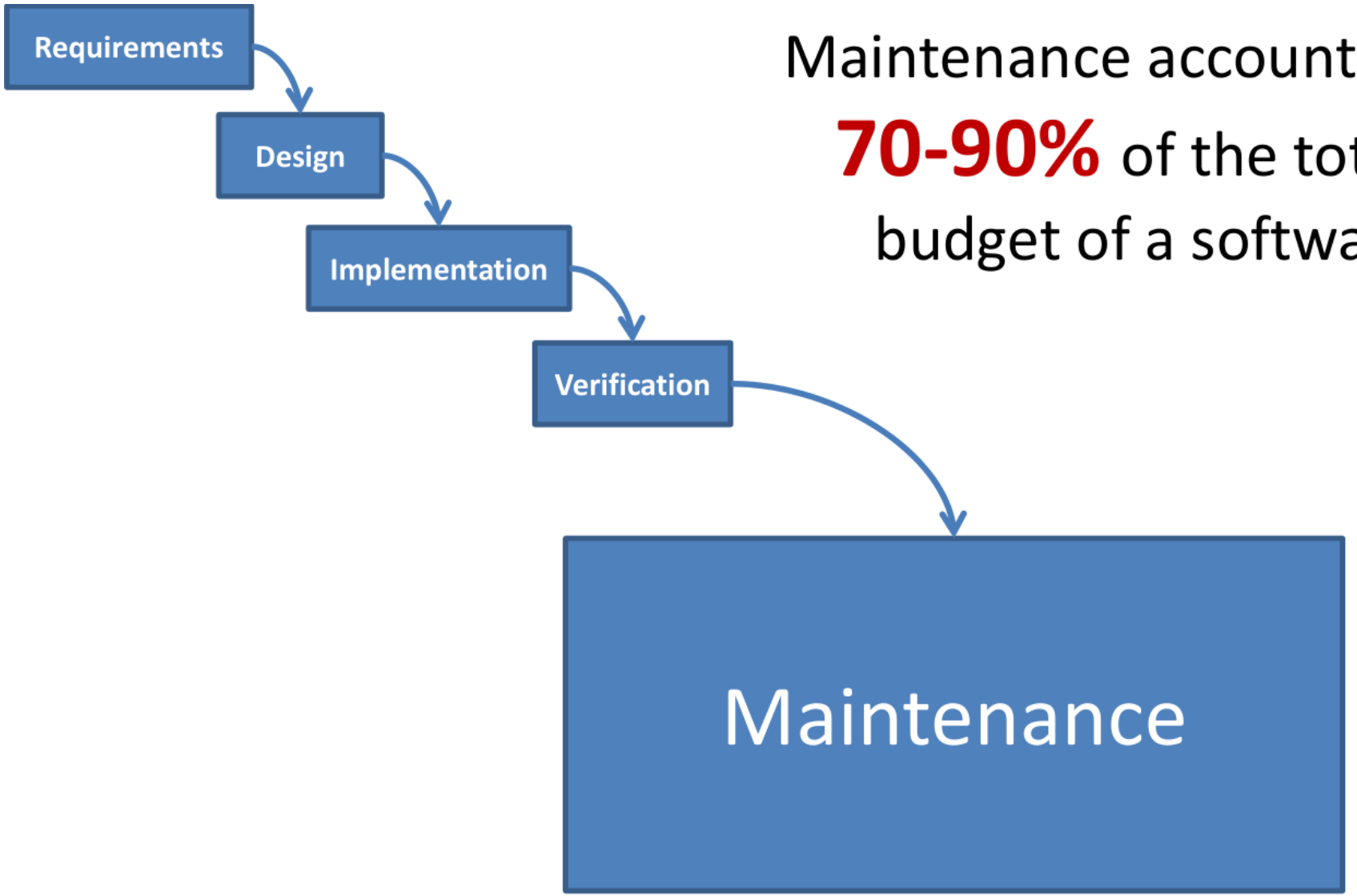


Figure 1. Tasks by time for *each* subject, normalized by the total time + time where events overlapped in each observation. Total observation time in hours is listed in parentheses after each subject's identification letter.

- Hour 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.]



Maintenance accounts for about **70-90%** of the total lifecycle budget of a software project.

[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.

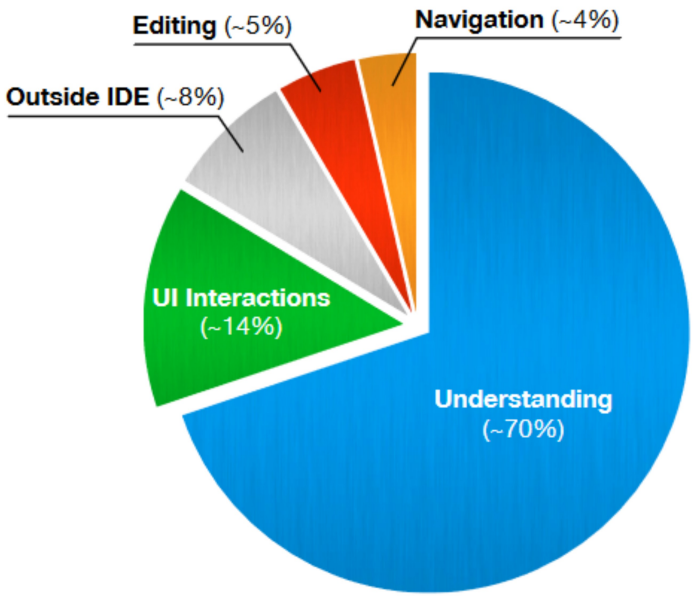
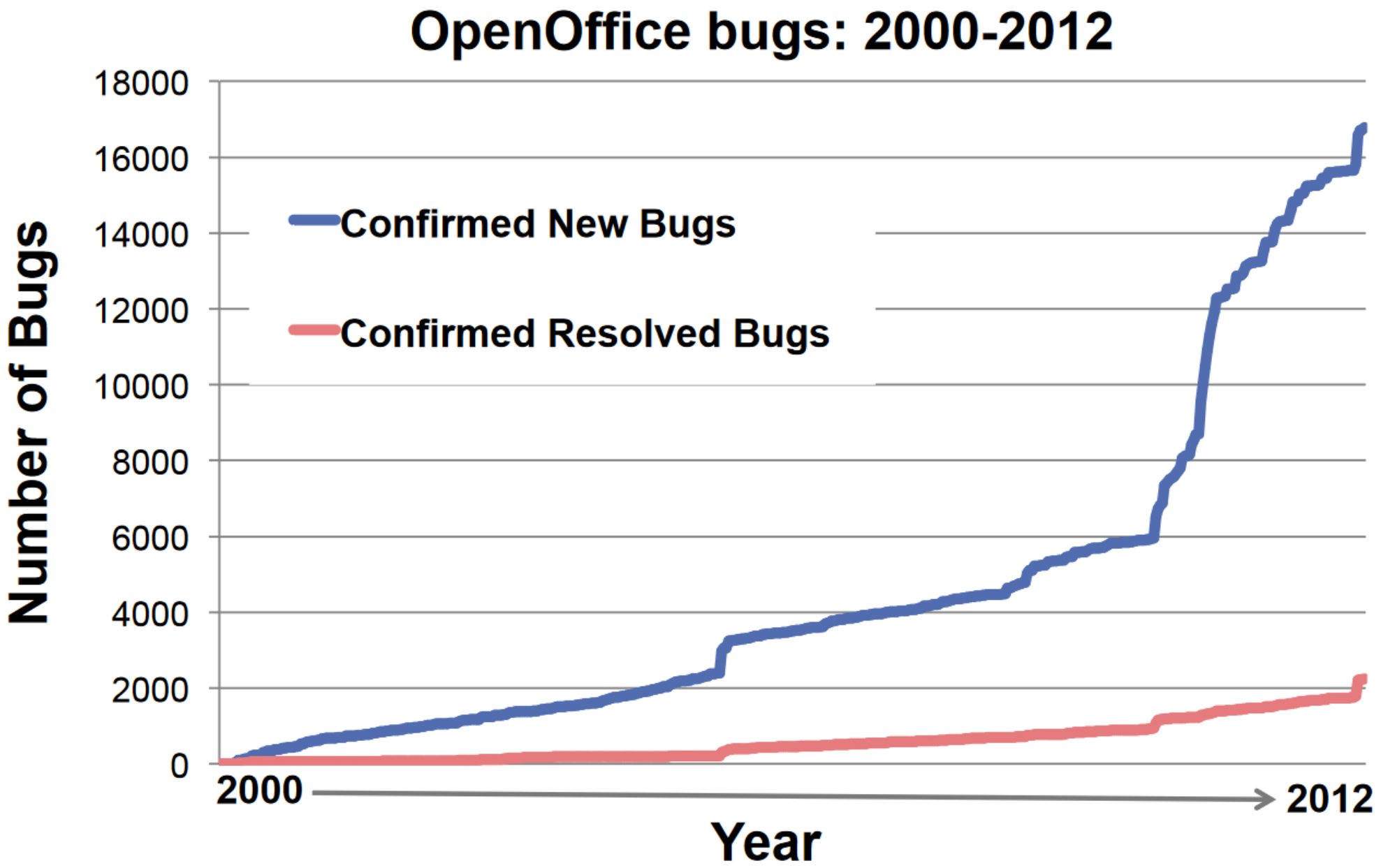


Fig. 3. How do developers spend their time?

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.]

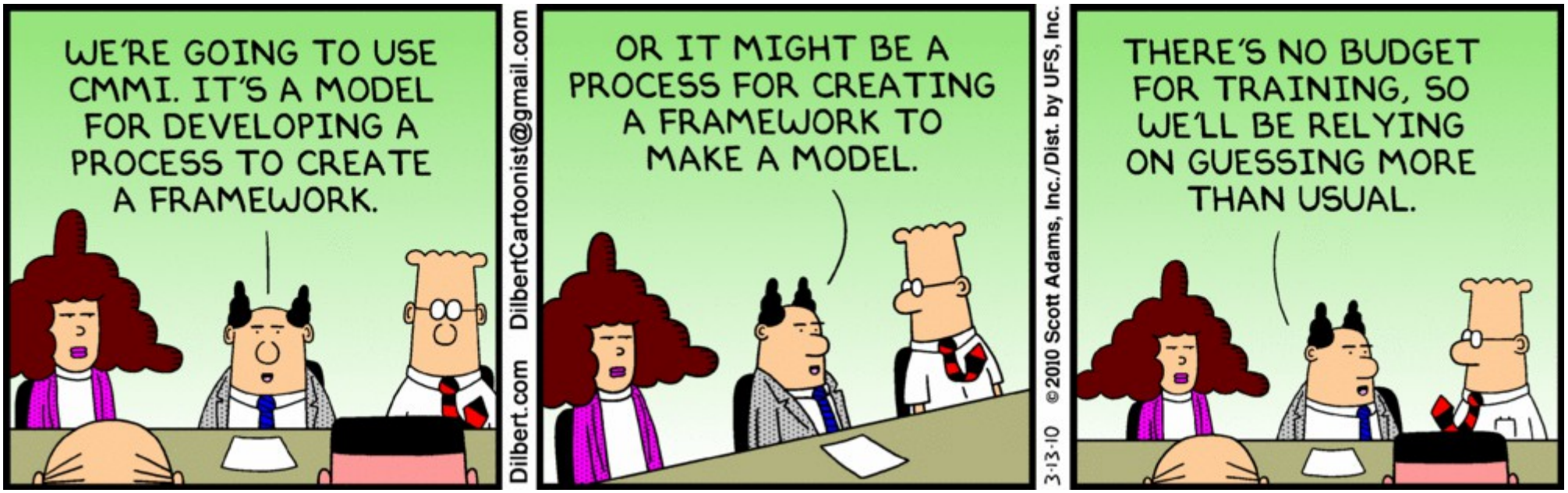


[Fry. *Leveraging Light-Weight Analyses to Aid Software Maintenance*. 2014]

A Key Issue

"Half of software engineering is crap."

– Your Instructor

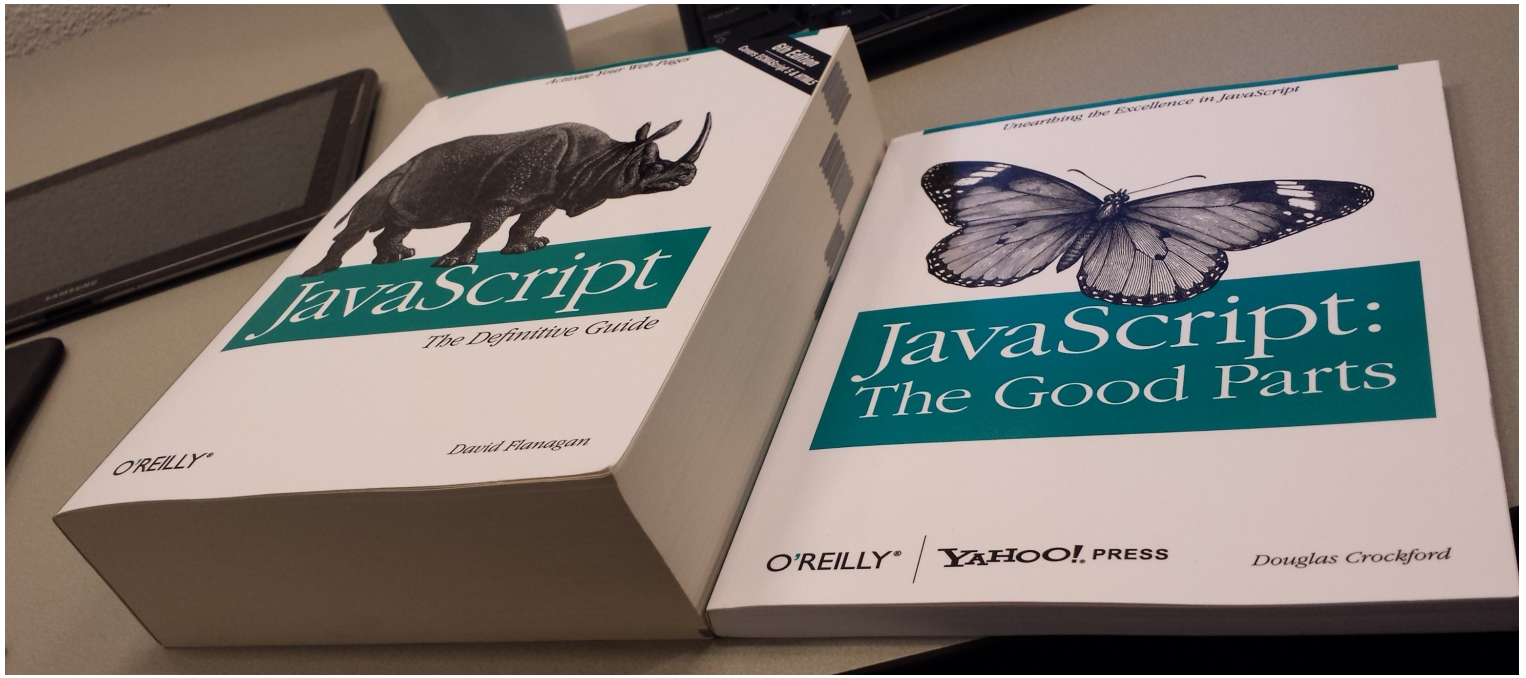


Revolutionary Solution

"So, by a vote of 8 to 2 we have decided to skip the Industrial Revolution completely, and go right into the Electronic Age."

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 2021?

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

- http://web.eecs.umich.edu/~weimerw/481/

- Administrivia
- Assignments and Grading
- Outline of Topics

We Want To Be Here! (WW)

- Have your professors worked in industry?
- Green Hills Software (private mid-sized, worked on debuggers for embedded systems)
- Sun Microsystems (large company, now Oracle, worked on circuit compiler and simulator)
- Microsoft (large company, worked on tools to find bugs in software)
- Also academia: works on advancing software quality, manages research group, 4x “10-Year Most Influential” research papers

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.

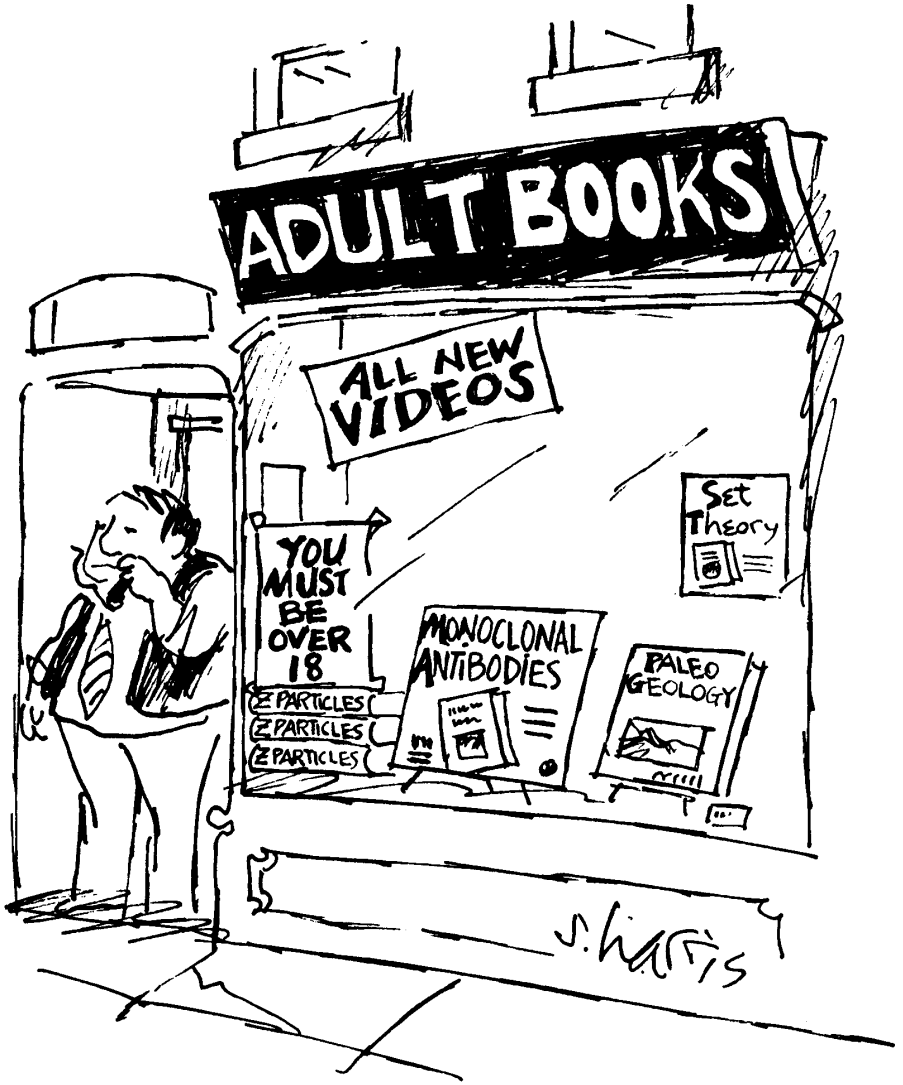
How will this help me graduate?

- Upper-Level CS/CE Technical Elective
- 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 mandatory
- 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 method […] 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 EECS usual

M Jan 10	Process, Risk and Scheduling [overview]	• Wikipedia's *Software Development Process* (Introduction and Section 3, Methodologies) • Buse and Zimmermann's *Information Needs for Software Development Analytics* (if you are low on time, read only Sections I-V) [Microsoft] • HW1 Specification • Optional: "How well do real companies plan and estimate effort?" Find out in: Anda et al.'s *Variability and Reproducibility in Software Engineering: A Study of Four Companies that Developed the Same System* [Simula]

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 your partner disappears
  - Use the forum to find partners, etc.

Lecture Time Slot Structure

- The last time 481 was offered to 300 students was purely remotely under COVID
  - There was one lecture and you either watched it live or watched the recording
- This time we want to offer a bit more
  - There is still one lecture and you can either watch it live or watch it recorded
  - But you can choose between attending the live lecture slot (1:30) or a staffed lecture slot (3:00) for the recording and structured activities (cont'd)

Lecture Slot 1 vs. 2

- Live Lecture Slot (1:30 in 1013 DOW)
  - Attend lecture, answer questions, ask questions live, some brief think-pair-share activities, few coding activities
  - Participation check example: question notecards
- Recording Practice Slot (3:00 in 1610 IOE)
  - Watch lecture (1.25x?). Activity, such as: 10-minute “try it on a real website”, fill in the blanks on this note sheet to guide your watching, practice (roleplay) requirements elicitation, pair programming with partner on HW, etc.
  - Participation check example: sign-in sheet
- Live recordings are always available, practice activity instructions are always available: you never miss anything

Lecture Slot Considerations

- Can mix-and-match which you attend from day to day (regardless of where you are registered/waitlisted)
  - Some students are more comfortable being called on, others may prefer talking with talking with friends. Some students find live lectures engaging, others prefer to watch at their own pace. Some students have seen topic XYZ on an internship and would rather have a structured time to practice a skill.
  - We acknowledge some students have scheduling constraints and we sympathize. (That's always true in CSE, sadly. Ask me about enrollment.)
- Previous 300-person semesters were “be there live vs. (watch the recording + do nothing)”, this semester we are trying “be there live vs. (watch the recording + do something)” to give students more options and support.

Discussion Sections

- Homework help (!), exam preparation, explain difficult material, answer questions
- Discussion sections
  - Half rephrasing lecture, hints on homeworks, etc.
  - Half office hours (we know you can attend them)
- Online OH will use Queue
- No required attendance
- Attend any: mix/match

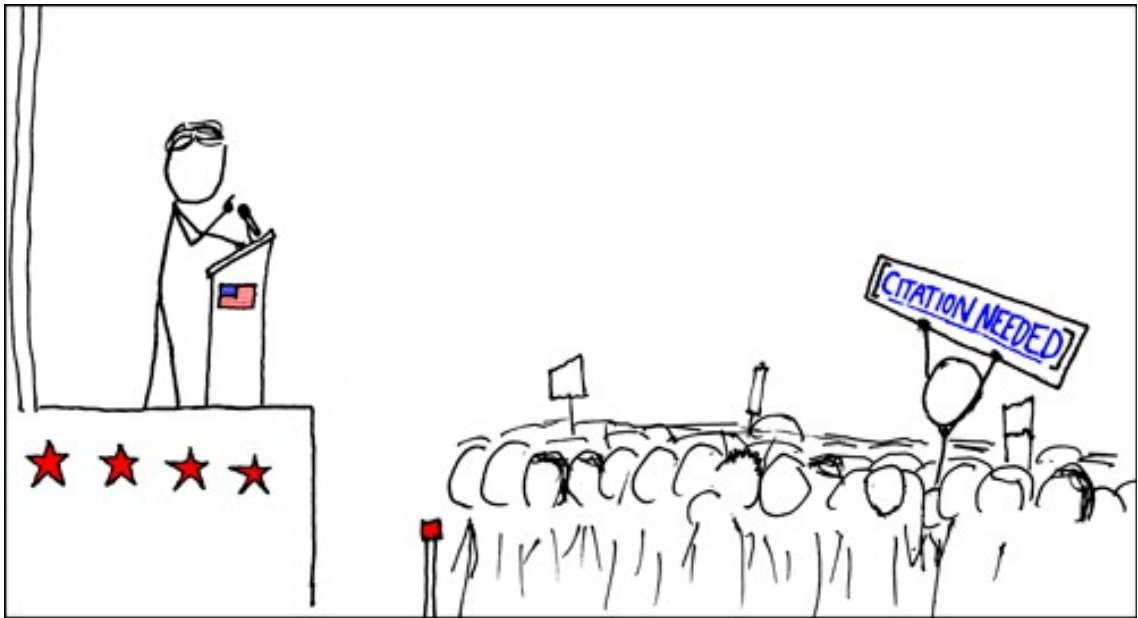
CATALIN PIT 💡🚀
@catalinmpit

Let me break it for you.

Coding is:
- 1% actually coding
- 40% debugging
- 15% coffee breaks
- 30% googling errors
- 9% staring with your colleagues at the screen
- 5% trying copy/pasted solutions from Stack Overflow

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

- "[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.]

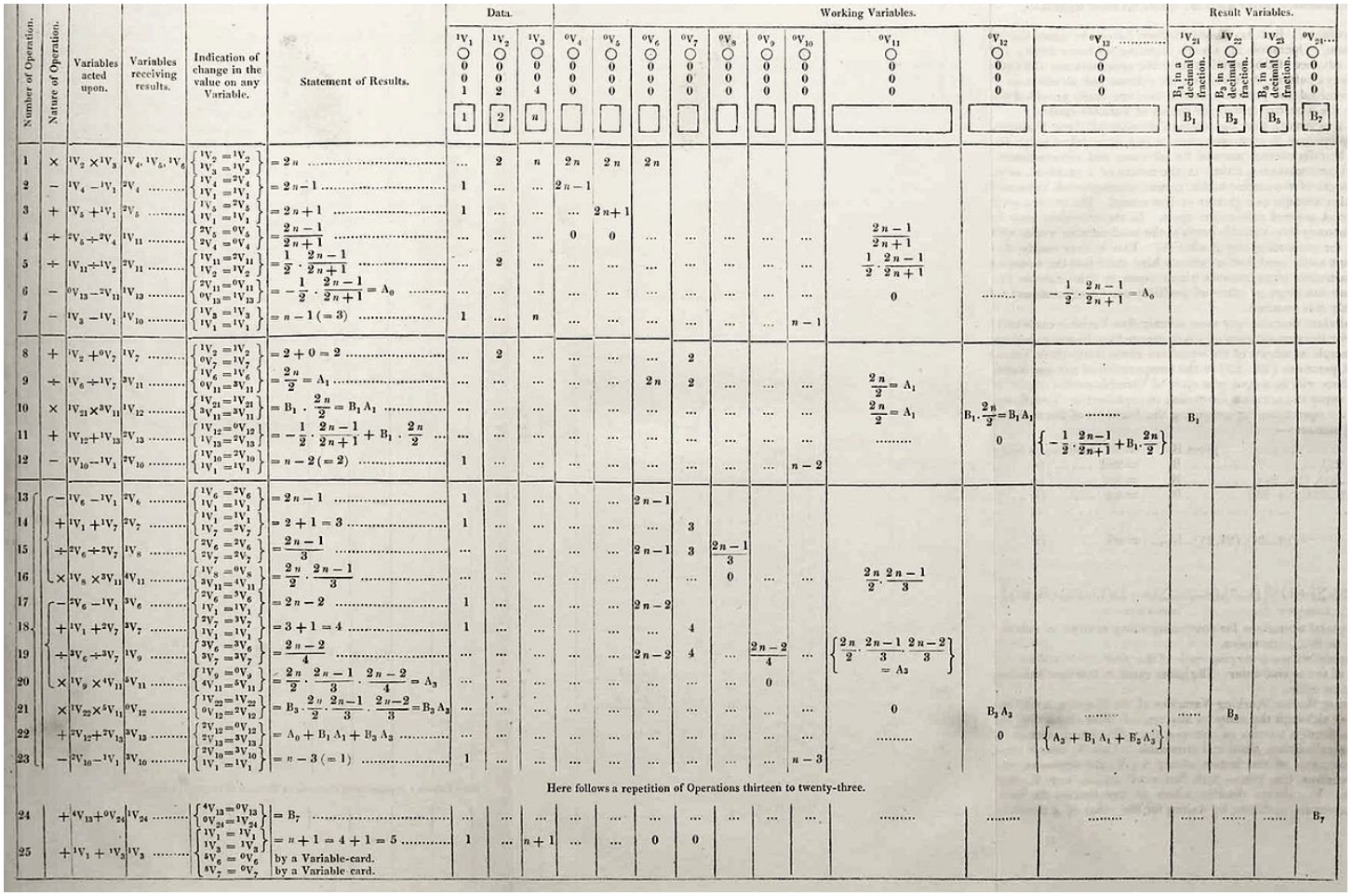
[Middendorf and Kalish. *The "Change-up" in Lectures.* TRC Newsletter, 8:1 (Fall 1996).]

Computer Science

- *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

- *What* did that first program do?



Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data. $^1V_1$ 1	Data. $^1V_2$ 2	Data. $^1V_3$ n	Working Variables. $^0V_4$	$^0V_5$	$^0V_6$	$^0V_7$	$^0V_8$	$^0V_9$	$^0V_{10}$	$^0V_{11}$	$^0V_{12}$	$^0V_{13}$	Result Variables. $^1V_{21}$ B_1	$^1V_{22}$ B_2	$^1V_{23}$ B_5	$^0V_{24}$... B_7
1	×	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$\left\{\begin{matrix} {}^1V_2 = {}^1V_2 \\ {}^1V_3 = {}^1V_3 \end{matrix}\right\}$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	−	$^1V_4 - {}^1V_1$	$^2V_4$	$\left\{\begin{matrix} {}^1V_4 = {}^2V_4 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n-1$	1	...	...	$2n-1$													
3	+	$^1V_5 + {}^1V_1$	$^2V_5$	$\left\{\begin{matrix} {}^1V_5 = {}^2V_5 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n+1$	1	...	...	...	$2n+1$												
4	+	$^2V_5 \div {}^2V_4$	$^1V_{11}$	$\left\{\begin{matrix} {}^2V_5 = {}^0V_5 \\ {}^2V_4 = {}^0V_4 \end{matrix}\right\}$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$						
5	+	$^1V_{11} \div {}^1V_2$	$^2V_{11}$	$\left\{\begin{matrix} {}^1V_{11} = {}^2V_{11} \\ {}^1V_2 = {}^1V_2 \end{matrix}\right\}$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\frac{1}{2} \cdot \frac{2n-1}{2n+1}$						
6	−	$^0V_{13} - {}^2V_{11}$	$^1V_{13}$	$\left\{\begin{matrix} {}^2V_{11} = {}^0V_{11} \\ {}^0V_{13} = {}^1V_{13} \end{matrix}\right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	.....	$-\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - {}^1V_1$	$^1V_{10}$	$\left\{\begin{matrix} {}^1V_3 = {}^1V_3 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n-1\ (=3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	+	$^1V_2 + {}^0V_7$	$^1V_7$	$\left\{\begin{matrix} {}^1V_2 = {}^1V_2 \\ {}^0V_7 = {}^1V_7 \end{matrix}\right\}$	$= 2+0=2$	...	2	...	...	...	...	2										
9	÷	$^1V_6 \div {}^1V_7$	$^3V_{11}$	$\left\{\begin{matrix} {}^1V_6 = {}^1V_6 \\ {}^0V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2} = A_1$						
10	×	$^1V_{21} \times {}^3V_{11}$	$^1V_{12}$	$\left\{\begin{matrix} {}^1V_{21} = {}^1V_{21} \\ {}^3V_{11} = {}^3V_{11} \end{matrix}\right\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	.....	B_1			
11	+	$^1V_{12} + {}^1V_{13}$	$^2V_{13}$	$\left\{\begin{matrix} {}^1V_{12} = {}^0V_{12} \\ {}^1V_{13} = {}^2V_{13} \end{matrix}\right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$ ...	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{-\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}\right\}$				
12	−	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$\left\{\begin{matrix} {}^1V_{10} = {}^2V_{10} \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n-2\ (=2)$	1	...	...	...	...	...	...	...	...	$n-2$							
13	−	$^1V_6 - {}^1V_1$	$^2V_6$	$\left\{\begin{matrix} {}^1V_6 = {}^2V_6 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n-1$	1	...	...	...	...	$2n-1$											
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$\left\{\begin{matrix} {}^1V_1 = {}^1V_1 \\ {}^1V_7 = {}^2V_7 \end{matrix}\right\}$	$= 2+1=3$	1	...	...	...	...	...	3										
15	÷	$^2V_6 \div {}^2V_7$	$^1V_8$	$\left\{\begin{matrix} {}^2V_6 = {}^2V_6 \\ {}^2V_7 = {}^2V_7 \end{matrix}\right\}$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$									
16	×	$^1V_8 \times {}^3V_{11}$	$^4V_{11}$	$\left\{\begin{matrix} {}^1V_8 = {}^0V_8 \\ {}^3V_{11} = {}^4V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2} \cdot \frac{2n-1}{3}$						
17	−	$^2V_6 - {}^1V_1$	$^3V_6$	$\left\{\begin{matrix} {}^2V_6 = {}^3V_6 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 2n-2$	1	...	...	...	...	$2n-2$											
18	+	$^1V_1 + {}^2V_7$	$^3V_7$	$\left\{\begin{matrix} {}^2V_7 = {}^3V_7 \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= 3+1=4$	1	...	...	...	...	...	4										
19	÷	$^3V_6 \div {}^3V_7$	$^1V_9$	$\left\{\begin{matrix} {}^3V_6 = {}^3V_6 \\ {}^3V_7 = {}^3V_7 \end{matrix}\right\}$	$= \frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{\frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = A_3\right\}$						
20	×	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$\left\{\begin{matrix} {}^1V_9 = {}^0V_9 \\ {}^4V_{11} = {}^5V_{11} \end{matrix}\right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0	...							
21	×	$^1V_{22} \times {}^5V_{11}$	$^0V_{12}$	$\left\{\begin{matrix} {}^1V_{22} = {}^1V_{22} \\ {}^0V_{12} = {}^2V_{12} \end{matrix}\right\}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...	0	$B_3 A_3$	.....	.....		B_3	
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$\left\{\begin{matrix} {}^2V_{12} = {}^0V_{12} \\ {}^2V_{13} = {}^3V_{13} \end{matrix}\right\}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{A_3 + B_1 A_1 + B_3 A_3\right\}$				
23	−	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$\left\{\begin{matrix} {}^2V_{10} = {}^3V_{10} \\ {}^1V_1 = {}^1V_1 \end{matrix}\right\}$	$= n-3\ (=1)$	1	...	...	...	...	...	...	...	...	$n-3$							
Here follows a repetition of Operations thirteen to twenty-three.																						
24	+	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$\left\{\begin{matrix} {}^4V_{13} = {}^0V_{13} \\ {}^0V_{24} = {}^1V_{24} \end{matrix}\right\}$	$= B_7$	...	...	...	...	...	...	...	...	...	...	.....	.....	.....	.....	.....	.....	B_7
25	+	$^1V_1 + {}^1V_2$	$^1V_3$	$\left\{\begin{matrix} {}^1V_1 = {}^1V_1 \\ {}^1V_2 = {}^1V_2 \\ {}^4V_5 = {}^0V_5 \\ {}^4V_7 = {}^0V_7 \end{matrix}\right\}$	$= n+1 = 4+1 = 5$ ..... by a Variable-card. by a Variable-card.	1	...	$n+1$	...	...	0	0										

Who Cared?
What was one "killer app"?

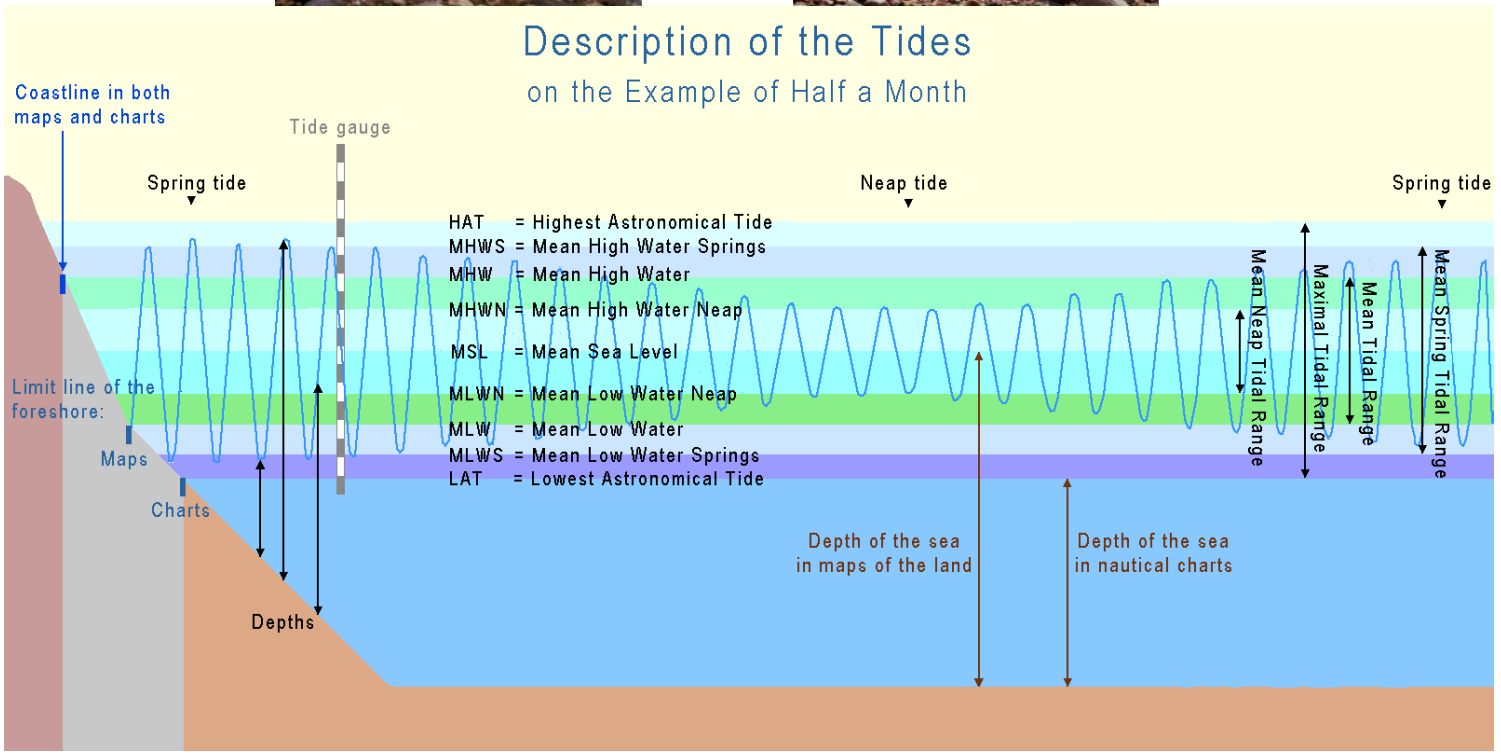
A hundred years later, the early Victorians used not Newton's theory of the tides but the variant that Bernoulli worked out for this competition, which they referred to as the equilibrium theory. The equilibrium theory outlined the essential mathematical formulas needed to predict the tides for each day of the year based on the positions of the sun and moon. The most important variable for each port was the "vulgar establ-

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

mer,'" 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

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

“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 choose 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

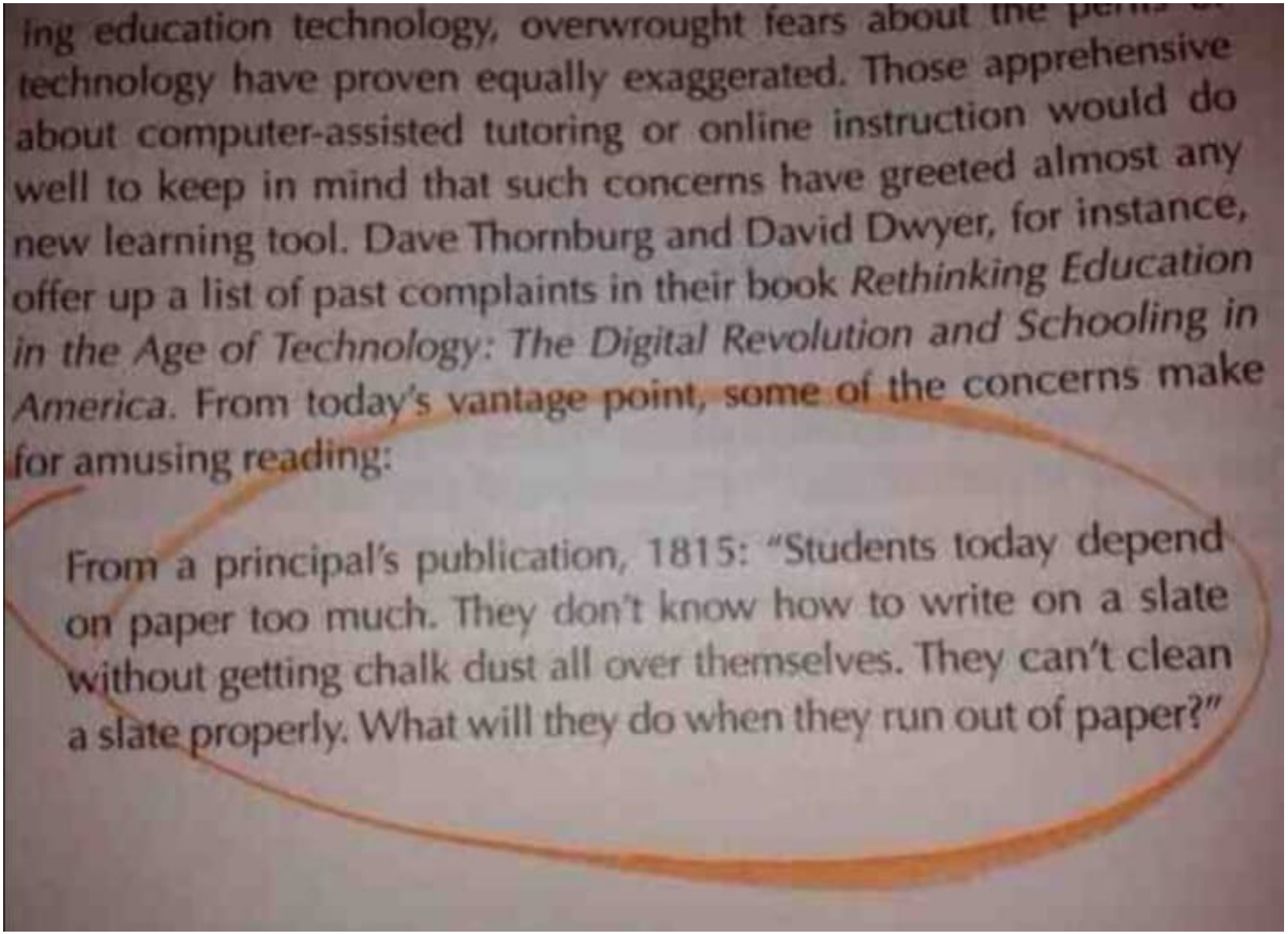
- Experimental findings:
  - Even when they knew the position came from a coin toss, subjects rated pro-Castro essay writers as having a positive Castro attitude.
  - “The subjects were unable to properly see the influence of the situational constraints placed upon the writers; they could not refrain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

Blah blah laptops blah …

(up hill 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 a 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

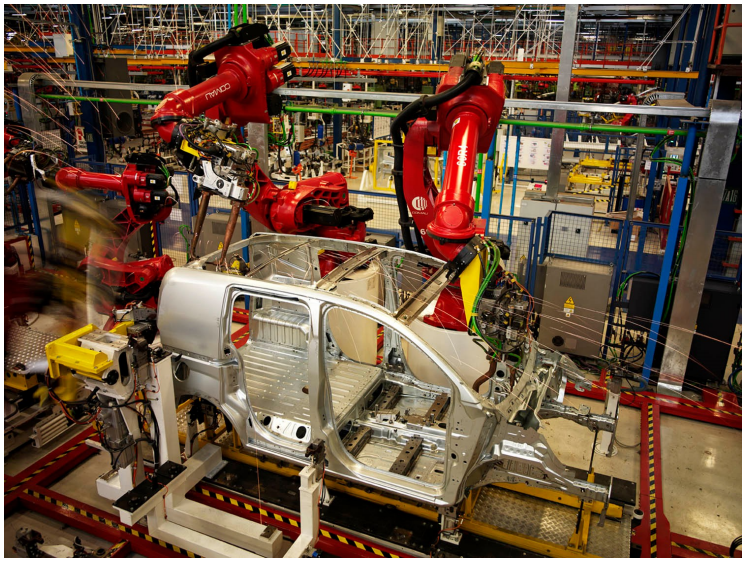
- Measurement and Risk
  - Process, scheduling, and information
- **Quality Assurance**
  - Code review, **testing**, and analysis
- Software Defects
  - Reporting and localizing
- Software Design
  - Requirements, patterns, and maintainability
- Productivity at Scale
  - People, teams, interviews, synthesis,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

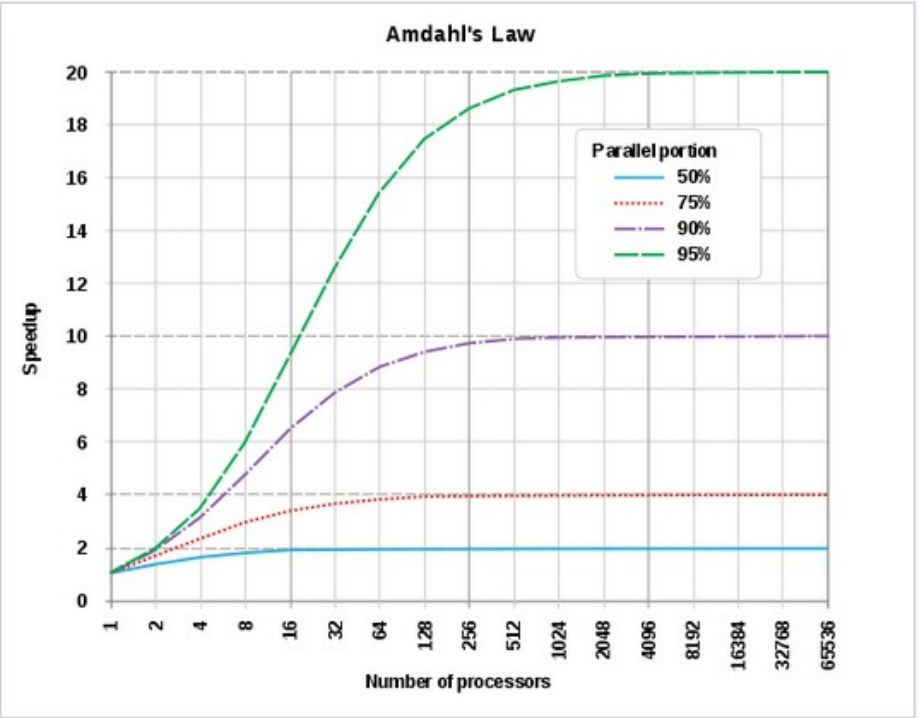
“My favorite operational definition of engineering is '**design under constraint**.' Engineering is creating, designing what can be, but it is constrained by nature, by cost, by concerns of safety, reliability, environmental impact, manufacturability, maintainability, and many other such 'ilities.'”

[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



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.

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

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

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

- “[a 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 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