Software Engineering

Our project plan will follow the usual arc.

Phase one will be unwarranted optimism supported by delusions of competence.

In phase two, the obstructionists will slither out of their lairs and try to smother our dreams.

Ignorance and envy will fuel rumors that get repeated until they morph into common knowledge.

Resources will be allocated based on misinformation and favoritism.

And requirements will drift until the project is both undesirable and impossible.

That brings us to the second week.

I want my unwarranted optimism back.
### List of public corporations by market capitalization

This list is up to date as of December 31, 2018. Indicated changes in market value are relative to the previous quarter.

<table>
<thead>
<tr>
<th>Rank</th>
<th>First Quarter</th>
<th>Second Quarter</th>
<th>Third Quarter</th>
<th>Fourth Quarter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Apple Inc.</td>
<td>Apple Inc.</td>
<td>Apple Inc.</td>
<td>Microsoft</td>
</tr>
<tr>
<td></td>
<td>▼851,317</td>
<td>▲909,840</td>
<td>▲1,091,000</td>
<td>▼780,520</td>
</tr>
<tr>
<td>2</td>
<td>Alphabet Inc.</td>
<td>Amazon.com</td>
<td>Amazon.com</td>
<td>Apple Inc.</td>
</tr>
<tr>
<td></td>
<td>▼715,404</td>
<td>▲824,790</td>
<td>▲976,650</td>
<td>▼748,680</td>
</tr>
<tr>
<td>3</td>
<td>Microsoft</td>
<td>Alphabet Inc.</td>
<td>Microsoft</td>
<td>Amazon.com</td>
</tr>
<tr>
<td>4</td>
<td>Amazon.com</td>
<td>Microsoft</td>
<td>Alphabet Inc.</td>
<td>Alphabet Inc.</td>
</tr>
<tr>
<td>5</td>
<td>Tencent</td>
<td>Facebook</td>
<td>Berkshire Hathaway</td>
<td>Berkshire Hathaway</td>
</tr>
<tr>
<td>6</td>
<td>Berkshire Hathaway</td>
<td>Tencent</td>
<td>Facebook</td>
<td>Facebook</td>
</tr>
<tr>
<td>7</td>
<td>Alibaba Group</td>
<td>Alibaba Group</td>
<td>Alibaba Group</td>
<td>Tencent</td>
</tr>
<tr>
<td>8</td>
<td>Facebook</td>
<td>Berkshire Hathaway</td>
<td>Tencent</td>
<td>Alibaba Group</td>
</tr>
<tr>
<td>9</td>
<td>JPMorgan Chase</td>
<td>JPMorgan Chase</td>
<td>JPMorgan Chase</td>
<td>Johnson &amp; Johnson</td>
</tr>
<tr>
<td>10</td>
<td>Johnson &amp; Johnson</td>
<td>ExxonMobil</td>
<td>Johnson &amp; Johnson</td>
<td>JPMorgan Chase</td>
</tr>
</tbody>
</table>
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 redistribute 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.”

Software is Critical: Driving

To specialize, 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."

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

OpenOffice bugs: 2000-2012

confirmed new bugs

confirmed resolved bugs

Number of Bugs

Year

[Conf. Leveraging Light-Weight Analyses to Aid Software Maintenance. 2014]
A Key Issue

“Half of software engineering is crap.”

- Your Instructor
"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
This Course

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

- Administrivia
- Assignments and Grading
- Outline of Topics
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
Blah blah laptops blah ...

(up hill both ways ...)

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.”

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.”

Assignments and Grading

- Assigned reading due before each lecture
- Attend lecture, take notes, visit forum

- Six homework assignments (~60%)
- Reading quizzes (~ 5%)
- Two examinations (~35%)

- See webpage for regrade and makeup policy
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

Mon Feb 05: Static Analysis Tools
- Wikipedia’s Static Program Analysis
- Ayewah et al.’s Experiences Using Static Analysis to Find Bugs [Google]
- Optional: “How does Microsoft find bugs in critical device drivers?” Find out in: Ball et al.’s A Decade of Software Model Checking with SLAM [Microsoft]
Assignments

• Six Assignments
  • 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: canvas or 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.
Discussion Sections

• Homework help (!), exam preparation, explain difficult material, answer questions
  • Discussion 1 — F 9:30-10:30am 1005 DOW, starting January 18th (led by Sabrina Benge)
  • Discussion 2 — F 10:30-11:30am 1018 DOW, starting January 18th (led by Sabrina Benge)
  • Discussion 3 — F 1:30-2:30pm 1014 DOW, starting January 18th (led by James Perretta)

• I don't care which one you attend
  • ... assuming there is space in the room, etc.
Software Engineering You Can Believe In

- Citations for strong claims (or ask on forum)
- Guest Lectures
  - Large companies, startups, researchers, etc.
- Readings from Industry
- Material from
  - Prem Devanbu
  - Christian Kästner
  - Marouane Kessentini
  - 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”


[ 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?
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 estab-

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

“Amazon Prime” circa 1842

Description of the Tides
on the Example of Half a Month

- HAT = Highest Astronomical Tide
- MHWS = Mean High Water Springs
- MHW = Mean High Water
- MHWN = Mean High Water Neap
- MSL = Mean Sea Level
- MLWN = Mean Low Water Neap
- MLW = Mean Low Water
- MLWS = Mean Low Water Springs
- LAT = Lowest Astronomical Tide

Depth of the sea in maps of the land
Depth of the sea in nautical charts
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.”


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

• No discussion sections or office hours this week!