CSE 110 – Software Engineering

Instructor: Professor Michael Coblenz
mcoblenz@ucsd.edu

Why Software Engineering?

Have You Ever…

- Submitted a project **late** or with **bugs** that you couldn't fix in time?
  - …bugs that you **didn't know about** before submitting?
- Written code that was **too hard to debug** or **maintain**, and wanted to **toss it out the window** at the end of the project?
- Wondered how people work on the same program for years without making a **mess**?

Building Great Software Is Hard

2/3 of projects are *late* [Tata]

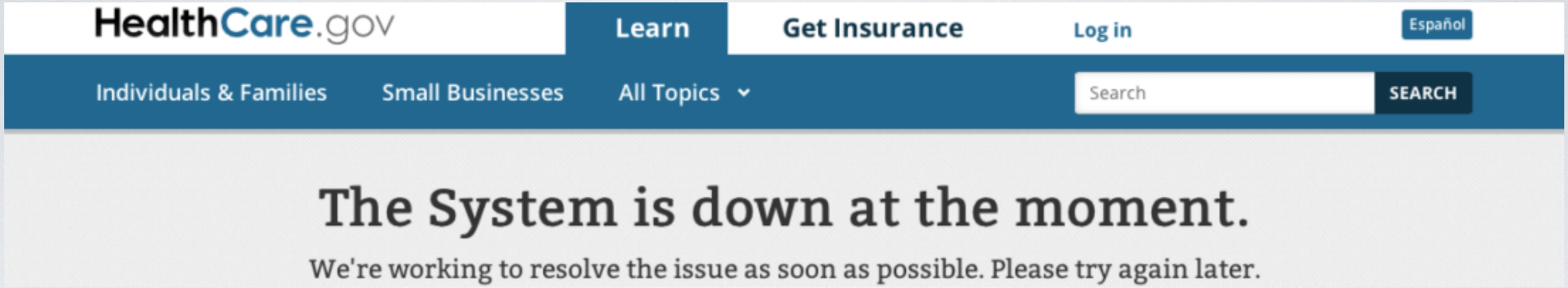
Allstate insurance planned a 5-year, $8M project. Six years later they replanned for $100M.

1/4 of all projects are *canceled* [Standish]

1/2 run *over budget* [Tata, SGR CACM]

http://www.galorath.com/wp/software-project-failure-costs-billions-better-estimation-planning-can-help.php

HealthCare.gov



- Demand (5x expected) took site down within 2 hrs. of launch
- Site incomplete (menus missing options, incomplete data transmitted to insurance companies)
- 6 users bought insurance the first day

https://d3.harvard.edu/platform-rctom/submission/the-failed-launch-of-www-healthcare-gov/

Healthcare.Gov Failure Causes

- HHS staff lacked experience launching technology products
- Failure to divide responsibilities appropriately
- Schedule pressure: launched before ready

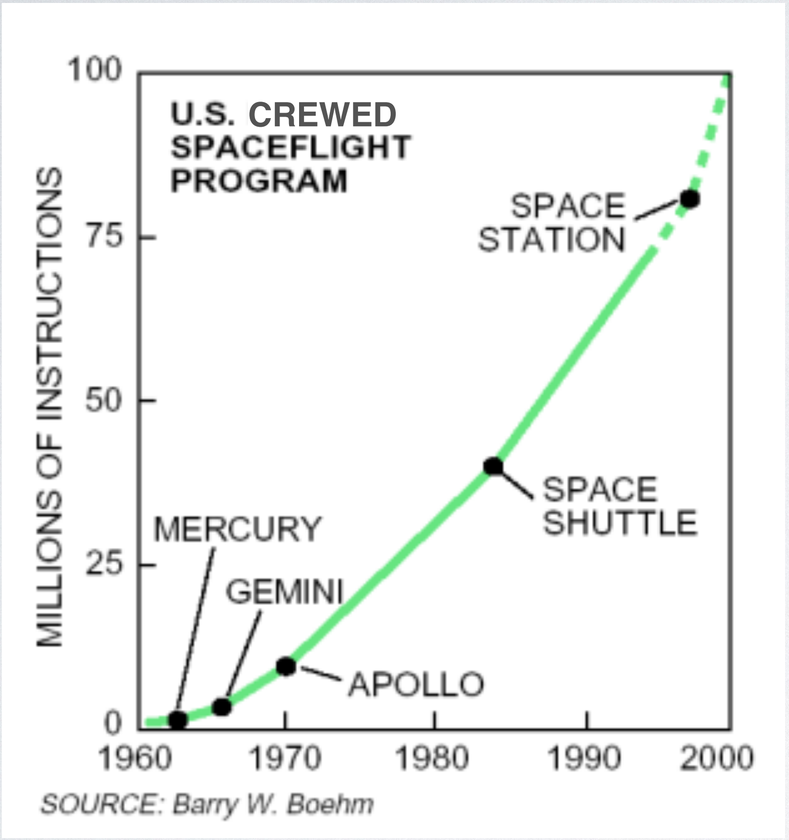
https://d3.harvard.edu/platform-rctom/submission/the-failed-launch-of-www-healthcare-gov/

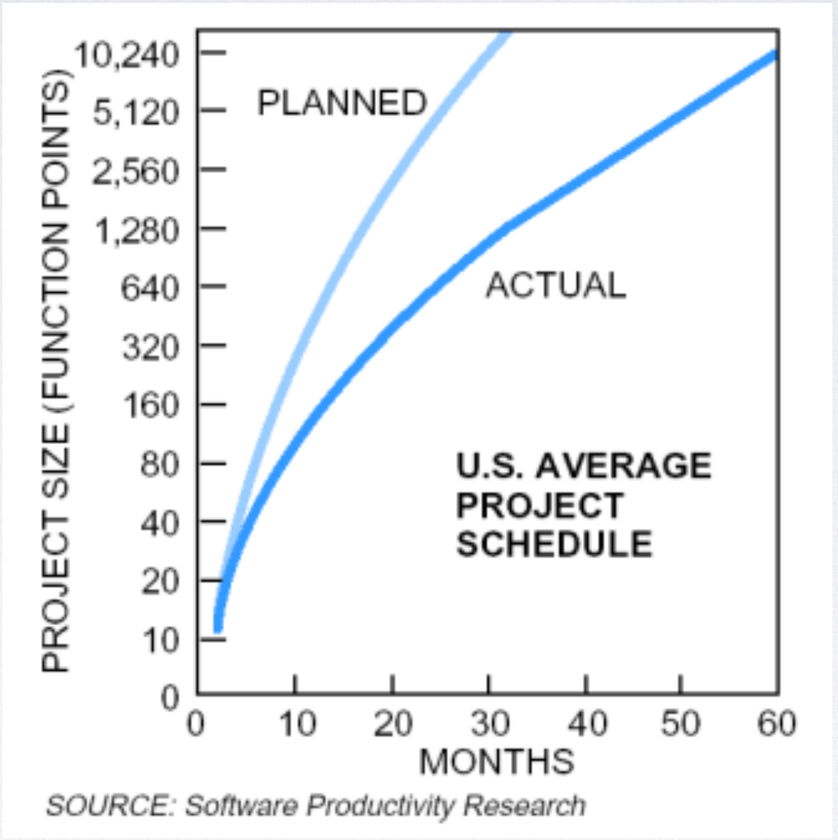
737 MAX

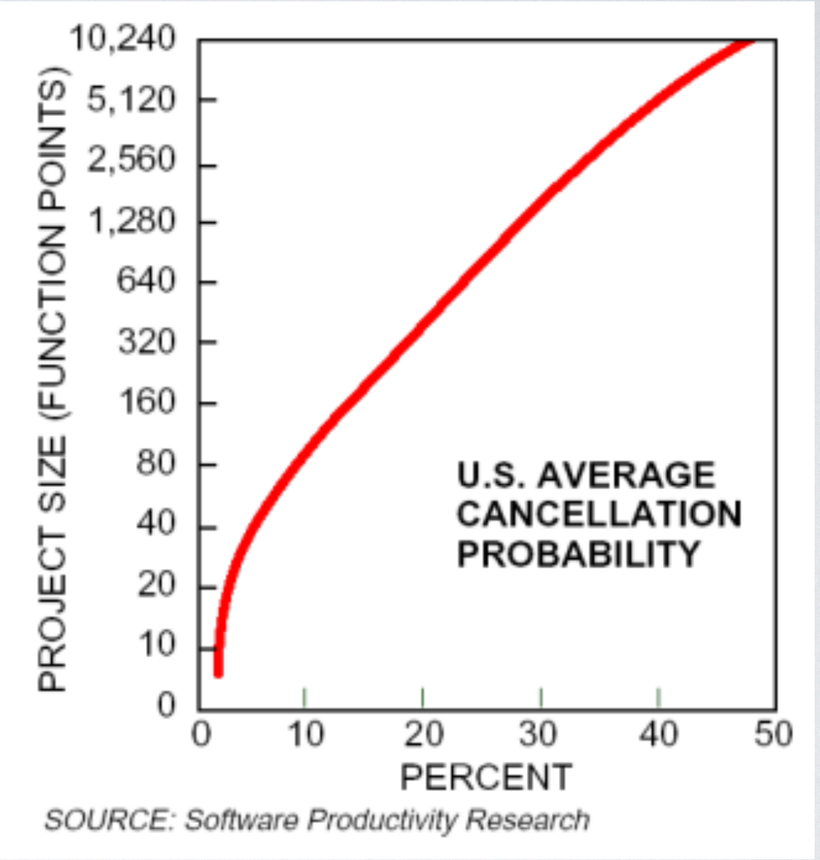
- To avoid cost of a major redesign, Boeing took shortcuts in aerodynamic design of 737 MAX
- Software was updated to compensate for side effects
- Software was not robust to angle of attack sensor failures (single point of failure)
- Pilots were insufficiently trained on failure modes
- Result: 346 deaths

https://spectrum.ieee.org/how-the-boeing-737-max-disaster-looks-to-a-software-developer

Why the Disasters? Scale.



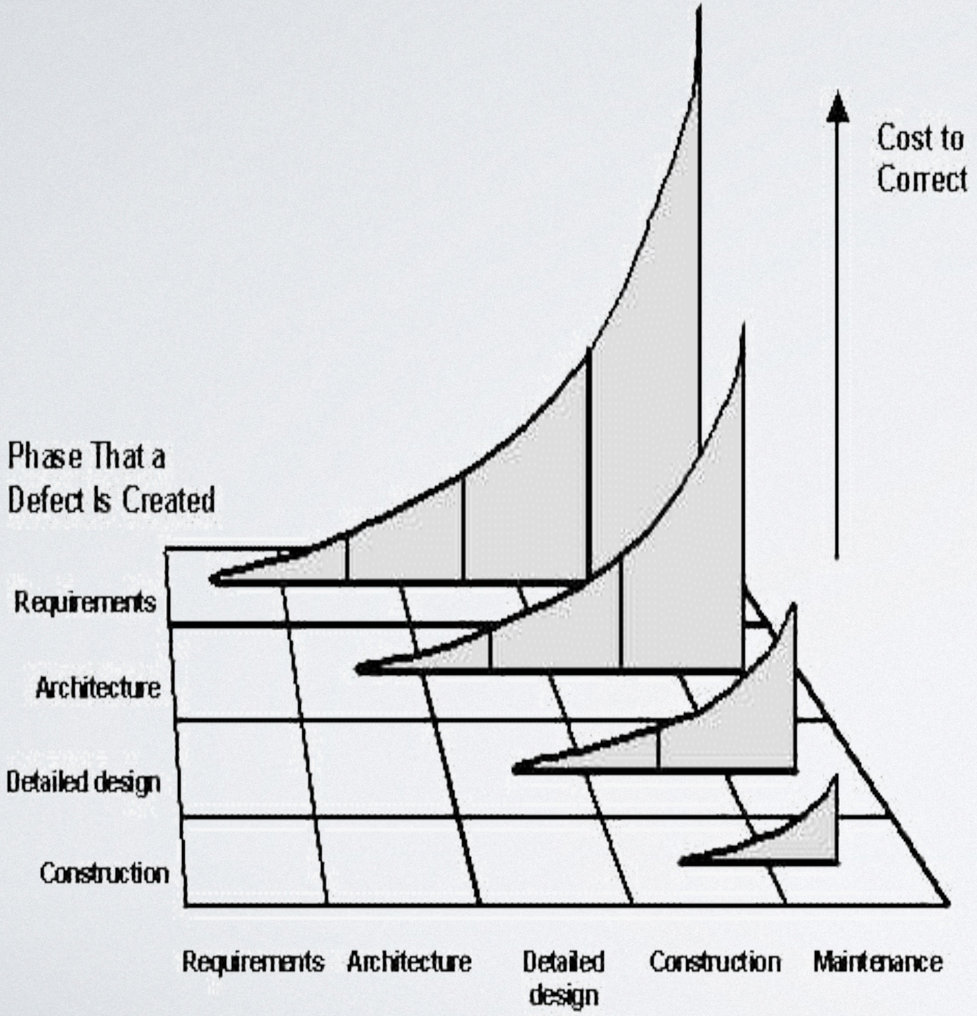




Users want more and more features

Gibbs, Software's Chronic Crisis, *Sci. Am.*, Sept. 1994

Why the Disasters?
Misunderstood and Changing Requirements



"…reworking a software requirements problem once the software is in operation typically costs 50 to 200 times what it would take to rework the problem in the requirements stage… A 1-sentence requirement can expand into…500 lines of code…and a few dozen test cases."

Steve McConnell, Software Quality at Top Speed,
Software Development, August 1996

Quality Control: a Short History

Quality control in early manufacturing was
Product-Centric ("what")

- Regularly test **product** outputs
- Make adjustments to factory as needed
- *But what to fix?*

mid-20th c., shift to **Process-Centric** ("how")

- Still test **product** outputs
- Also measure **process** elements
  - *plans, people, tools, product-in-progress*
- Use **cause-and-effect model** to adjust factory
- Statistics to precisely track variation
- Buzzword: *Statistical Process Control*

- **SE has inherited this legacy**
- ***SE methods are process-centric***

What's a Software Process?

How to produce quality software

- What: what customer wants, on time, under budget, free of flaws

Steps include:

- Planning
- Execution
- Measurement
  - Product, and process itself
  - Examples: bugs, progress, time, feature acceptance by customer

Technical Themes of the Course

- **Scale**
  - All of computer science, especially CS research, is about *managing scale*. So is SE.
- **Risk, Uncertainty**
  - SE is all about *managing risk*.
  - Goal: increase upside risk (great products), decrease downside risks (late, buggy, etc.)

Beyond Process

- Process is just the beginning
- Software engineering is about quality decision-making
  - Good architecture
  - Teamwork
  - Good design
  - Thorough quality assurance
- This course is about all of these things.

Secrets (To Be Revealed)

- Q: How do you build what is needed, and not just what you *think* is needed?
  - A: Iteration with frequent feedback from users.
- Q: When I change code, I almost always break something else. How do I avoid doing that?
  - A: Good design and architecture.
- Q: How do I make my software useful for speakers of other languages? And protect my users' data? And behave *ethically*? And build a team?
  - A: Keep coming to class.

This course will be different
from every other course you've ever taken!

Most Courses You've Taken Probably…

- Tell you what to build, prove, or write
  - If you build, prove, or write *that*, then you get an A.
- Expect correct answers
- These courses are great for teaching you how to build, prove, or wrote those kinds of things.

Most Courses You've Taken Probably…

- Focus on *individual* work
- Or, if there are groups, each group creates an artifact
  - and everyone is graded according to the artifact's quality

In **This** Course…

- **You** will decide what to create.
- If you learn *key principles*, *work hard*, and make *wise choices*, you will get an A.
- Often there will be multiple good answers (and you can get an A for any of them if you justify your choice well).

In **This** Course…

- Teamwork will be central to your success
  - But not everyone on the team will get the same grade
  - because each member will be graded on their *contributions*

In **Most** Project Courses

- More is better
  - More features
  - Better performance
  - More documentation
  - Longer, more detailed report

In **This** Course

- Wisdom is better

A Dilemma (On a Solo Project)

- You promised users you will deliver your software (a turboencabulator) in two days.
- Most of it works, but you have been debugging the hydrocoptic marzlevanes for a day now. There's no telling how long it'll take, but it could be 2-3 days more.
- There are assorted other bugs.
- Now what?

What Do?

A. Work very late, try to finish everything and fix the bugs

B. Ask VP (nicely) to assign more resources (another engineer)

C. Fix bugs first and see whether there's time left for the hydrocoptic marzlevanes

D. Cut hydrocoptic marzlevanes; focus on quality

E. Something else

Goals of the Course

- Work effectively in a team that uses an Agile development process
- Design and document software systems according to stakeholder needs
- Implement and debug complex software systems
- Bottom line: able to think in terms of **tradeoffs** and **risks**

About Me

- Research: making software engineers more effective, mostly via better programming languages
- Recent work: REST API design; Rust language; programming for scientists; debugging research
- Previously: Senior Software Engineer at Apple (eight years)

TAs

- Tony Li: PhD student, five years of professional software experience (e.g., Facebook)
- Ria Dharmani: MS student, two years at Amazon + Microsoft
- Jash Makhija: MS student, TAd 110 3x before, internships at JP Morgan, ASUS, etc.

Tutors

- Boyu Tian
- Can Yavuz
- Jonathan Zhang
- All have substantial internship experience, including with the tech stack you'll be using

About Class

- Discussion is an integral part of class!
  - Research has shown that *active learning* is superior
- BE HERE at 4 PM

Changes

- This is my first time teaching CSE 110!
- I have made significant changes compared to other quarters.
- Expect changes.
  - Assignments may appear or disappear; grade weights may change a little.
  - Policies may need to be adjusted.

Health

- Your health comes first
- Do not come to class sick
  - You can catch up on the podcast later
- Masking is currently optional
- Reminder: new COVID vaccine is available (+ flu vaccine)

A Dual Agenda

- Course design choice: learn **technologies** or **principles**?
- But: learning principles is best done **in context**
- Lecture will focus on principles
- Labs will focus on technologies
- Discussions will include **very important group activities** (see calendar)
  - Also: exam review and extended discussion

Grading

- Individual assignments: 25%
- Team project: 23% (everyone gets the same grade)
- **Personal contributions** to team success: 18%
- Labs: 4%
- Midterm: 10%
- Final: 20%

Team Project Details

- You will choose your own requirements!
- Every project will be different
- Your team will be assigned a TA, who will help you manage your project
- We will grade you according to ongoing quality and progress
  - **not** on final quality

Example

- Sam's team's project has been going great.
- In Week 6, the team realizes that the plan to use services from spurving.com was a bad one: not enough documentation, too many bugs, nowhere to get help. They replace it with grammeters.com, but this delays the project a week. They re-prioritize, dropping a key feature.
- In Week 7, three team members catch COVID and are out for a week.
- The team meets in Week 8 and cuts two features that key stakeholders had requested.
- The team demos successfully in Week 10 — without those features.

Assessment

- The project still meets the most critical user needs.
- The team made wise decisions, re-scoping the project as needed.
- The three team members who got sick don't contribute much that week, but since we drop the lowest week's contribution scores, it doesn't matter.
- Perhaps everyone gets an A.

Another Example

- Sam's team's project has been going great.
- In Week 6, the team runs into trouble: components relying on spurving.com aren't working.
- But spurving.com is very popular and must be great. Sam starts working very late.
- Week 7: three team members catch COVID and are out for a week.
- Week 8: not much works yet (ongoing trouble with spurving.com). Sam's not getting enough sleep to work effectively.
- Week 9: team members double down on their commitment to ship all their features — gotta keep the users happy. Sam skips doing homework for other classes to deliver the project.
- Week 10: demo goes poorly. The features are all "there" but each one exhibits serious bugs.

Assessment

- The team didn't reconsider their strategy after missing deadlines due to issues with spurving.com.
- The team didn't drop features despite resource constraints (three people sick), committing them to either poor quality or an unacceptable workload.
- Features don't help if they don't work or are unpolished.
- Team grade: not an A.

Aside

- A worse scenario
  - Three team members catch COVID
  - To avoid impacting the team, they continue working hard and coming to team meetings
  - Now everyone has COVID and no one is getting work done.

Personal Contributions to Team Success

- You will submit peer feedback (strengths, weaknesses) each week
- Then, TAs will give you in three areas each week:
  - Independence
  - Teamwork
  - Technical contributions

Teamwork

- Teamwork may be the hardest part of the class
- Team skills are a *learning goal*
- I, TAs, and tutors are available to help!
- Raise issues with each other and staff before they become serious, if possible
- Note: instructor and TAs are "responsible employees"
    - Please tell us about incidents of harassment, but know that we must report unlawful discrimination and harassment to OPHD

Project

- Being a student is stressful!
- Create something to help *some* UCSD students with stress.
- We will match you with about 6-8 students based on your interests, availability, and experience. Survey is out **now** (due Monday).

Key Project Steps

- You'll match with a team *next week* (even if you are on the wait list)
- **Focus groups** will provide insight regarding what **users** want
- A **vision document** will say what you plan to create
- A **mockup** will show the application in detail
- In **five sprints**, you will build your app.

Team Project

Component	Weight
Topic + requirements questions for focus groups	2%
Vision document	4%
Revised vision document	4%
Mockups	4%
Sprint 1 quality and progress	2%
Sprint 2 quality and progress	2%
Sprint 3 quality and progress	2%
Sprint 4 quality and progress	2%
Sprint 5 quality and progress	2%
Project demo video	1%

One More Thing…

- You wouldn't want to miss a final exam question, would you?
- You can buy *insurance* against this unfortunate event.
- Suppose you bomb a question about topic X. Oops.
- If you answered all the *insurance clicker questions* in the lecture on topic X correctly, you get 50% on the question!

Insurance

- Points are pro-rated according to the fraction of questions you got right (50% right gets you a 25% score on the question)
- Insurance applies to partial credit (if you earn 50% partial credit but have full insurance, you get 75%)
- If you miss class (up to twice, no explanation needed):
  - Watch the podcast; email me an explanation of how the material impacts your project. I will give credit if I am convinced you learned the material.

Insurance

- This is an EXPERIMENT
- If I catch a whiff of insufficient academic integrity, or there are negative impacts on the course, I will stop the whole scheme.
- Insurance questions will be marked with 
- No insurance offered today
- You have to trust me to be reasonable about mapping lectures to exam questions
- Insurance is ONLY for the final exam, not the midterm

Practice Question

Which best characterizes what you should do to maximize your grade in this course?

A. Grind hard, work late.

B. Make very complex apps that showcase your brilliance.

C. Make wise choices, manage risk, and work effectively with your team.

Questions About the Course?

What Do You Want To Learn?