CSE 210 – Principles of Software Engineering

Michael Coblenz
Computer Science & Engineering
UC San Diego
`mcoblenz@ucsd.edu`

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**

Introductions

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

About Class

- Discussion is an integral part of class!
    - Past attempts have shown: Zoom is not as good
- BE HERE at 9 AM
    - (sorry about the early start)
- To promote open discussion, class will NOT be recorded
- This is a new class (to me!)
    - Expect changes

Health

- Your health comes first
- Do not come to class sick
    - Instead, contact me for a Zoom link if you're up to it
- Masking is currently optional
- I will have masks available for those who want

Course Design

- Course design choice: learn **technologies** or **principles**?
- This class is optimized for learning **principles.**
- In assigning teams: we will assign according to the *tech stack* you want to learn and your schedule availability
- But we won't teach a specific technology
    - A quarter isn't enough anyway

Grading

- 50% contribution to your project team
    - Technical contributions
    - Teamwork
    - Independence/leadership
- 50% individual work

Individual assessment

- Reading responses
- Homework assignments
- Research has shown: when given feedback and grades, students focus on grades
  - But the feedback represents learning opportunities!
- Graded on "OK" or "resubmit" basis
- Complete all responses/assignments with "OK" for an A on individual work

Teamwork

- Teamwork may be the hardest part of the class
- Team skills are a *learning goal*
- I and TAs are available to help!
- I will adapt content according to challenges you have
- 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

Questions about the course?

Why software engineering?

Building Great Software is Hard

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

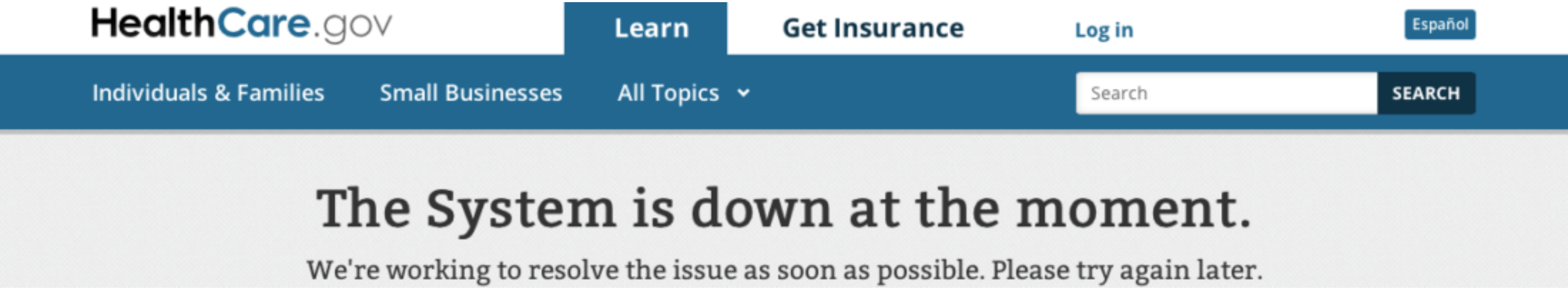
1/4 of all projects are *cancelled* [Standish]

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

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

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

Healthcare.gov



The System is down at the moment.

We're working to resolve the issue as soon as possible. Please try again later.

- 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

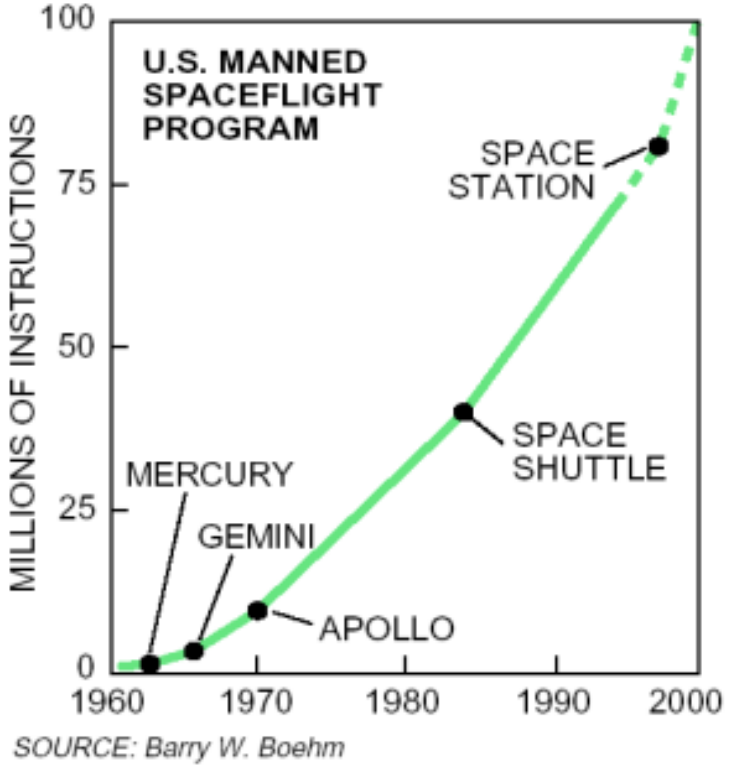
https://d3.harvard.edu/platform-rctom/submission/the-failed-launch-of-www-healthcare-gov/

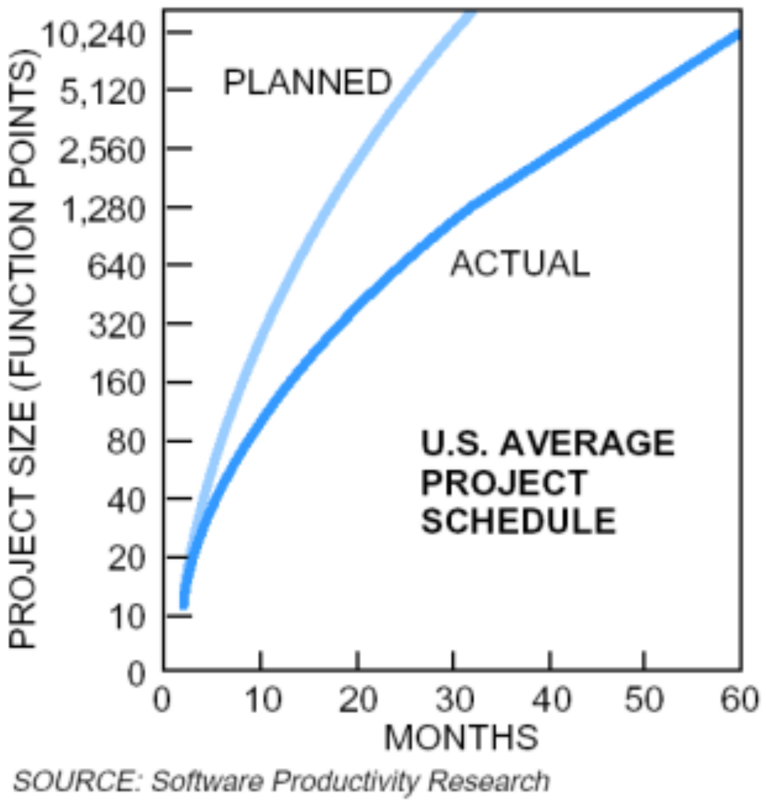
737 MAX

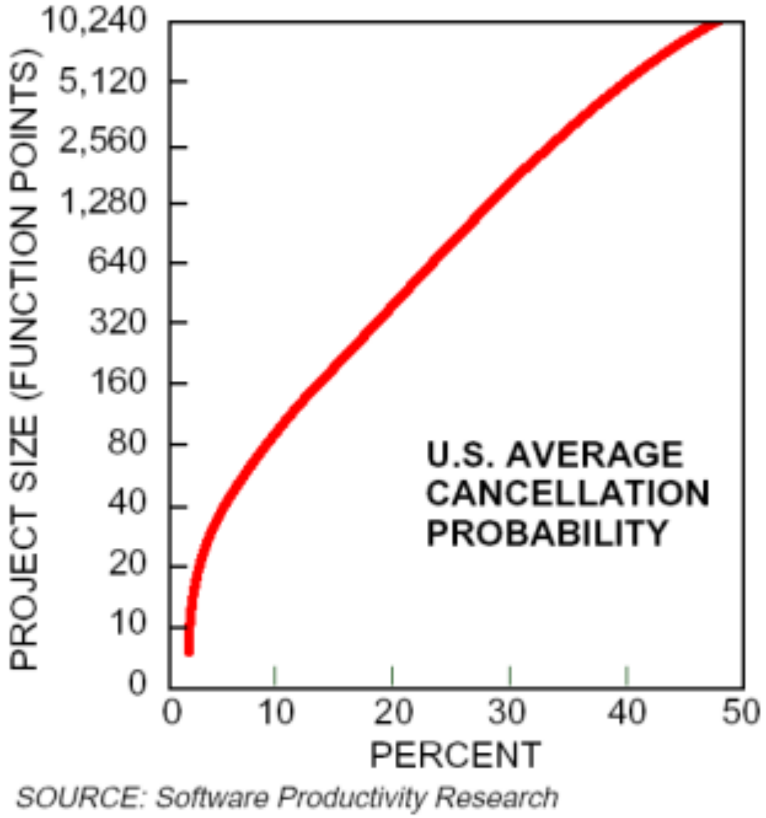
- To avoid cost of a major redesign, Boeing took shortcuts in aerodynamic design of 737 MAX
- Software was updated to compensate for side effects
- Software was not robust to angle of attack sensor failures (single point of failure)
- Pilots were insufficiently trained on failure modes
- Result: 346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

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

Why the disasters? Scale.



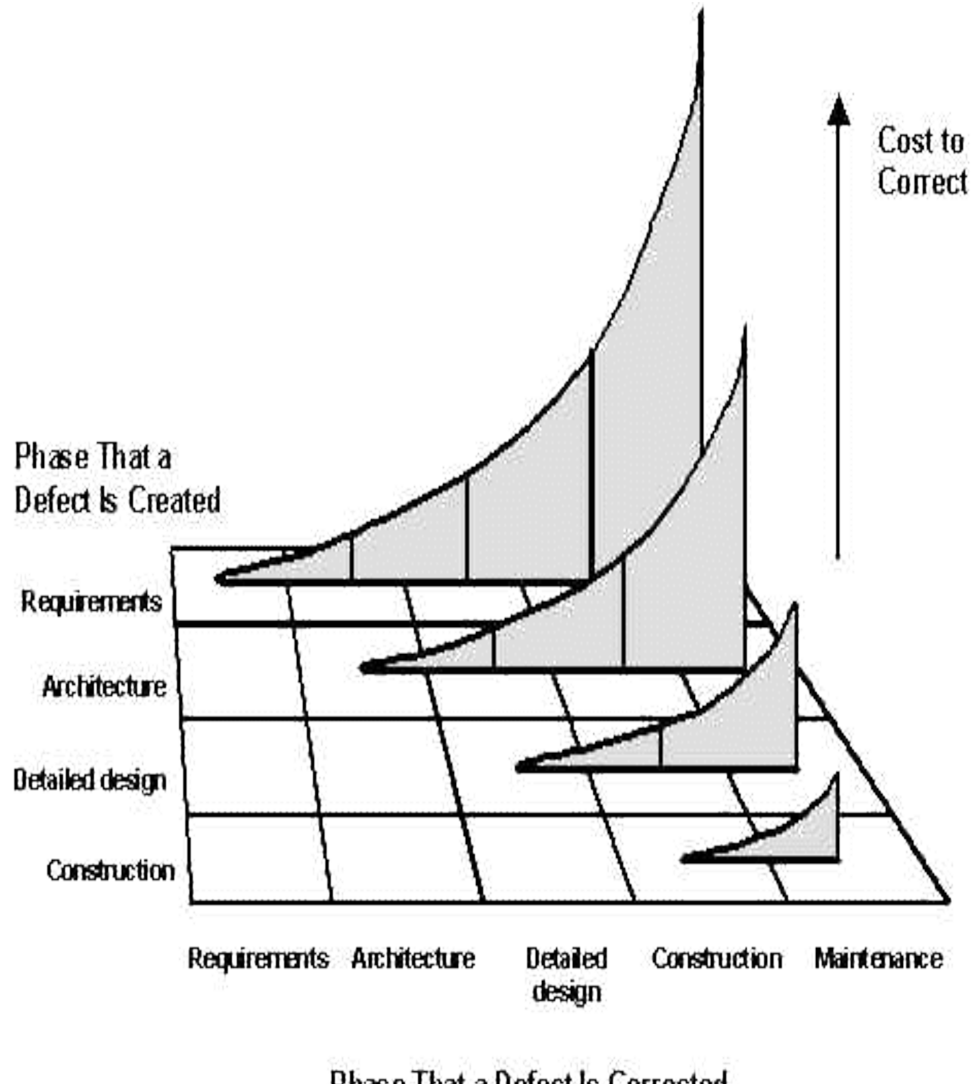




Users want more and more features

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

Change/Evolution yields Complexity/Bugs

Figure 4 Serial and average growth trends of a particular attribute

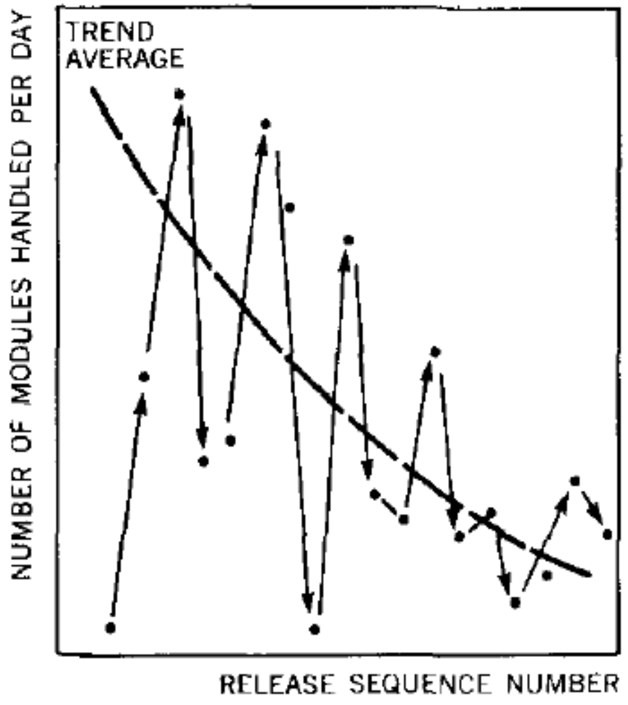
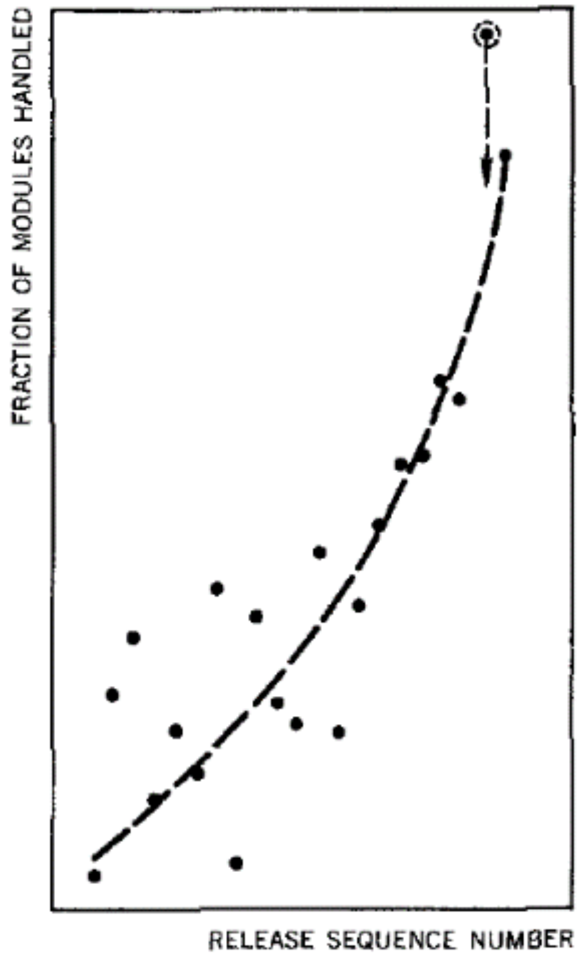
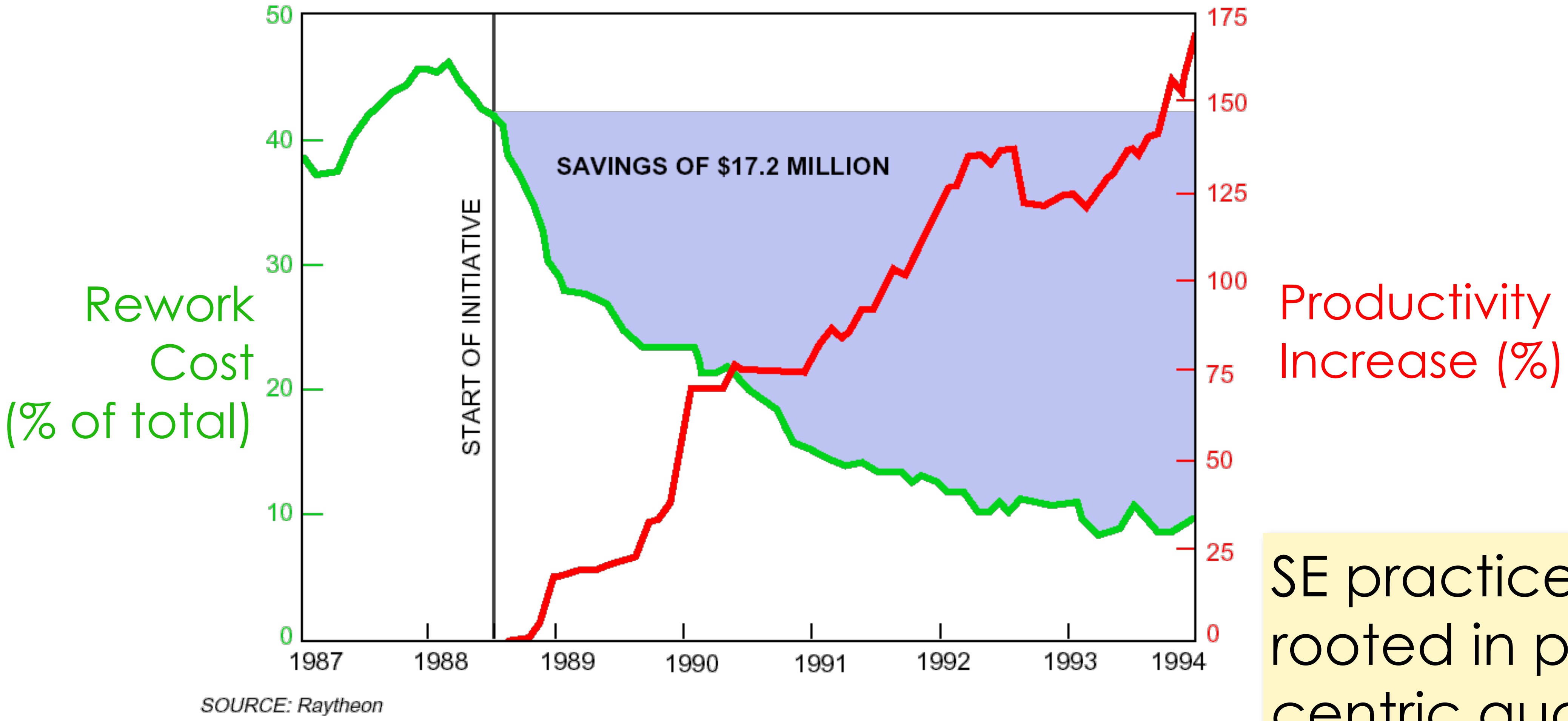


Figure 7 Complexity growth during the interval prior to each release



Belady & Lehman, A Model of Large Program Development, IBM Systems Journal, (15)3, 1976

S.E. Practices Like Agile Make a Difference



Results of Raytheon's use of best-practices.

SE practices are rooted in process-centric quality control

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 as needed
- Statistics to precisely track variation
- Buzzword: *Statistical Process Control*

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

What’s a Software Process?

It’s the “how” that produces the “what” – quality software

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

A prescribed sequence of steps

Steps include:

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

A software process is a self-aware algorithm

- *Observes and adapts according to measurements*

Agile processes are adaptive to the “customer”

- Features, schedule, budget, priorities, markets, change
- Must measure these as well as internal elements (correctness)
- Easily extended to adapting to many other “problems”
    - …as long as they can be observed and measured

The Changing Face of Software

Applications

- Web 2.0, Mobile 2.0, …
- Ubiquitous computing
- Developing world
- Big data, AI, ….

Methodologies

- Open Source
- Agile (XP, Scrum)

Technologies

- Web services, JavaScript, AJAX, JQuery, …
- Programming environments
- Component-based, Model-driven software development

Do we rewrite the rules,
or just reinterpret them?

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*. Doing something important requires taking risks. SE seeks to increase upside risk (great products), while decreasing 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.

Project

- Housing is tough in San Diego!
- Let's do something to make it better.
- Finding apartments?
- Landlord reputation?
- Finding roommates?
- Increasing transparency: utility costs? Environmental impact? Where can I get fiber?

First Week: Intro to Agile Process

You come from many backgrounds

Seen different variants of software process

I’m going to introduce a generic Agile Process

Will be point of contrast for much of course

Also will be used in project

Looks like a lot of reading, but actually not many words

- Don’t skip the side bars and pictures!
- Great examples, great exercises, Q&A
- Don’t have to do the crossword puzzles

What do you want to learn?