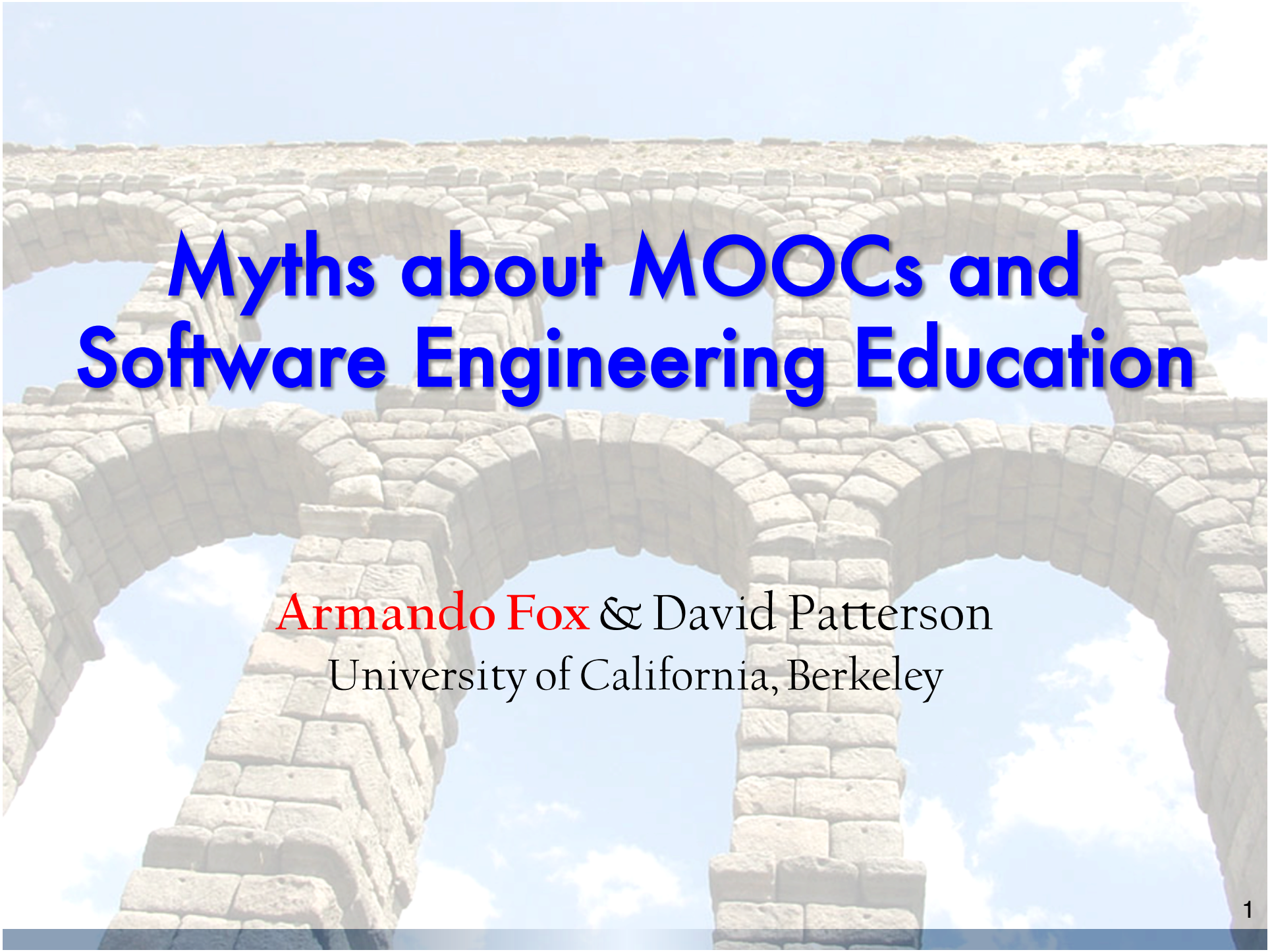


Myths about MOOCs and Software Engineering Education

Armando Fox & David Patterson
University of California, Berkeley

Outline

- Myths about Software Engineering Education
- Revamping SW Engineering: Agile, SaaS, and learning by doing
- Myths about MOOCs (Massive Open Online Course)
- Experience with SPOCs (Small Private Online Course, Scalable Personalizable Online Course)
- Conclusion: SPOC+Ebook == 21st Century Textbook (curricular technology transfer)

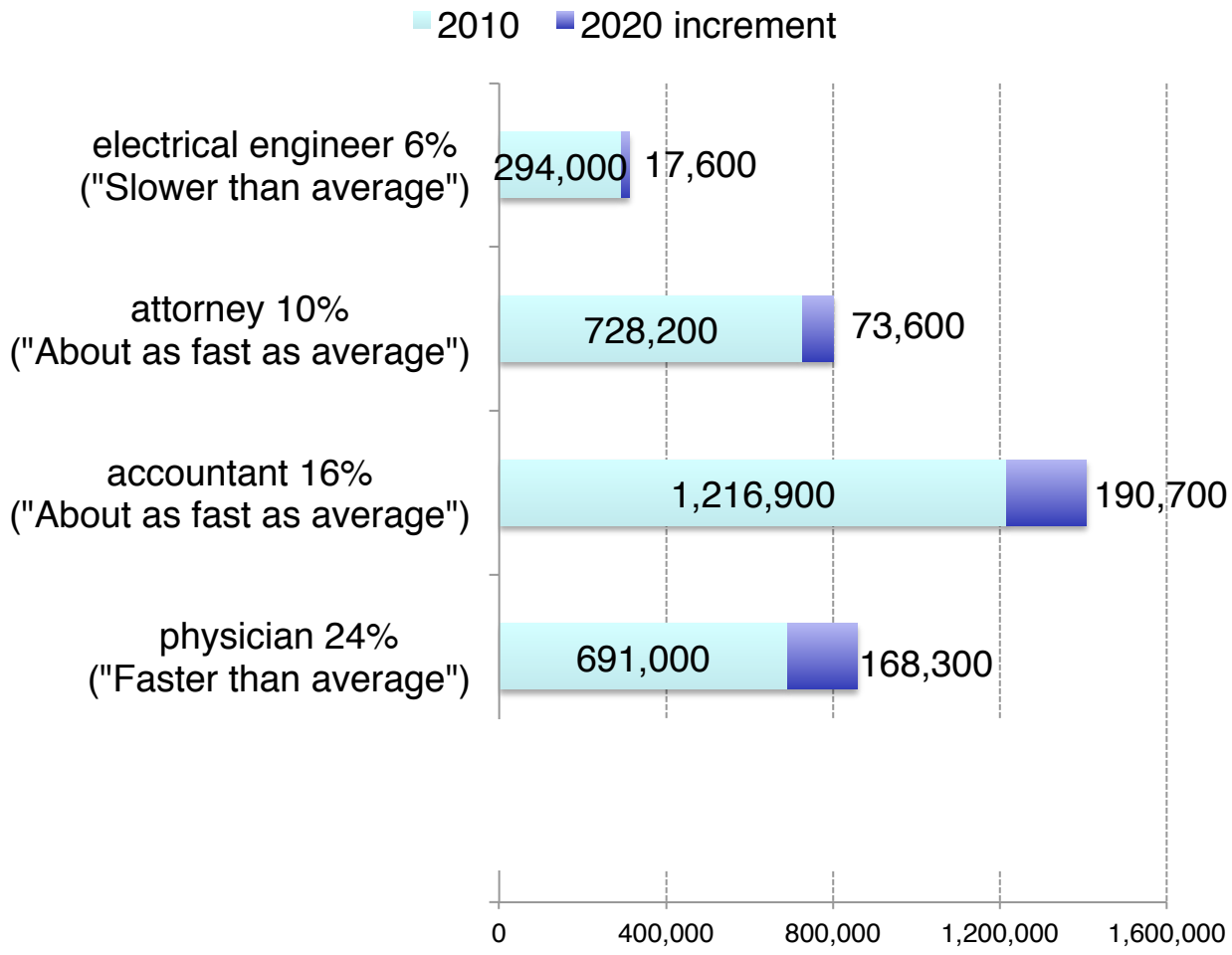
Myth 1: No SW Eng Jobs in US

"*I'm sick of hearing all the whining about how outsourcing is going to migrate all IT jobs to the country with the lowest wages. … Alas, high school students and their parents believe these rumors, and they are scared about majoring in computer science…[In fact, the] Office of Technology of the U.S. Department of Commerce reports that between 1999 and 2004, U.S. IT employment grew 17 percent.*"

David Patterson, "Stop whining about outsourcing!" *Queue* (2005)

Myth 1: No SW Eng Jobs in US

Dept. Labor % Increase & Number of jobs, 2010 vs. 2020 (as of 9/2013)



Myth 1: No SW Eng Jobs in US

1. Software Engineer

16. Petroleum Engineer

28. Civil Engineer

40. Physician

58. Nuclear Engineer

60. Aerospace Engineer

66. Mechanical Engineer

73. Electrical Engineer

83. Industrial Engineer

85. Receptionist

87. Attorney

104. Airline Pilot

137. High School Teacher

163. Police Officer

178. Actor

185. Firefighter

196. Newspaper Reporter

200. Lumberjack

Top Jobs 2012. Based on salary, stress levels, hiring outlook, physical demands, and work environment (www.careercast.com)

Myth 2: SW Eng == Programming

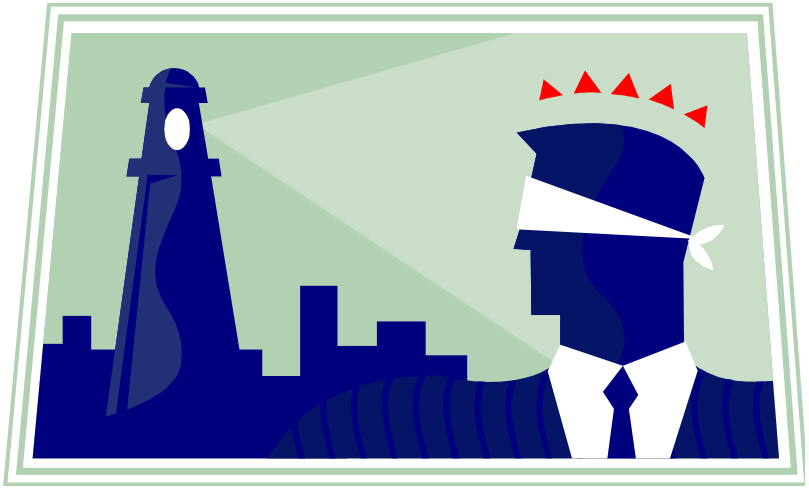
- Reality: Programmers Eternally Optimistic
  - Almost perfect code 1st time
  - Little debugging, then ready
  - Don't waste time with specs, testing framework, …
- SW Eng ≈ Vitamins
  - May be good for you, but optional

Myth 3: Enough time to teach SW Eng

- Reality: 4 to 5 full time weeks to teach SW Engineering!
- CS/CE Degree => 1 semester or 2 quarter courses
- Students take 4 courses:
  - 1 * 15 weeks / 4 ≈ 4 weeks
  - 2 * 10 weeks / 4 ≈ 5 weeks

Myth 4: Instructor is Expert

- Reality: SW Eng novices
- Unlikely faculty are practicing SW Engineers
- Unlikely faculty are even researchers in SW Engineering
  - 16 faculty taught UC Berkeley SW Eng course in last 20 years



Myth 5: SW Development method/process less important

- Reality: Many to choose from
- Plan-and-Document Methodologies:
  - ≈ Civil Engineering
  - Waterfall, Spiral, Rational Unified Process, …
- Agile Methodologies:
  - ≈ Movie Making
  - Extreme Programming, Kanban, Scrum, …

Myth 6: SW Eng Textbooks enable "learning by doing"

- Reality: surveys of methodologies, platforms, issues
  "I hear & I forget, I see & I remember, I do & I understand"
- Leading textbook in 7th Edition (1st Edition 1982)
-

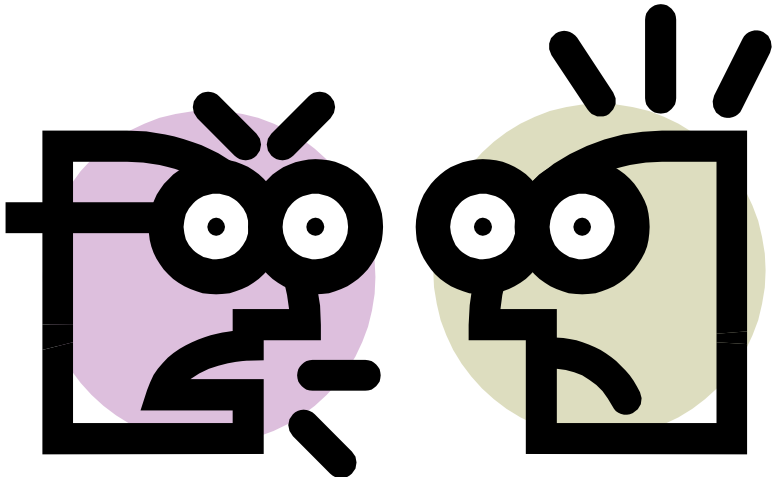
• *"This is just a horrible book and it's unfortunate that many CS students have to get stuck using it"*
• *"Horrible out-of-date Software Engineering book"*
• *"I found the book very hard to read, due in part to poor organization, writing style, and FLUFF!"*
• *"Train Wreck In Print"*
• *" Buy only if you want a narrow view of plenty of outdated topics"*
• *"This book is actively harmful to Software Engineering"*

Myth 7: SW Eng Tools Good for Classroom Use

- Reality: Many SW methodologies require tools too expensive or inappropriate for classroom
- So less likely for students to follow SWE guidelines
  - And harder for instructors to check if do follow them

Myth 8: SW Eng Course prepares students for careers

- Reality: Industry traditionally complains about SW Eng Courses at every university
- (Only CS course so widely defamed?)



Common Sad, Stable Situation

- Students ignore lectures, build SW as always have
  =›just a project course
  - frustrating to instructors
  - boring to students
  - disappointing to industry
- Reward for teaching SW Eng: poor evaluations
  - 16 faculty taught SWE last 20 years vs. 6 teach data structure or compilers

Outline

- Myths about Software Engineering Education
- Reinventing SW Engineering with Agile+SaaS
- Myths about MOOCs (Massive Open Online Course)
- Experience with SPOCs (Small Private Online Course)
- Conclusion: SPOC+Ebook == 21st Century Textbook (curricular technology transfer)

Revamping Berkeley Course

- Ask SW companies for advice
  - Amazon, eBay, Google, Microsoft, Salesforce, VMware
- Students can write code, but lack basic SW skills, especially:
  1. Dealing with legacy code (unanimous)
  2. Working in team for non-technical customer
  3. Automated testing

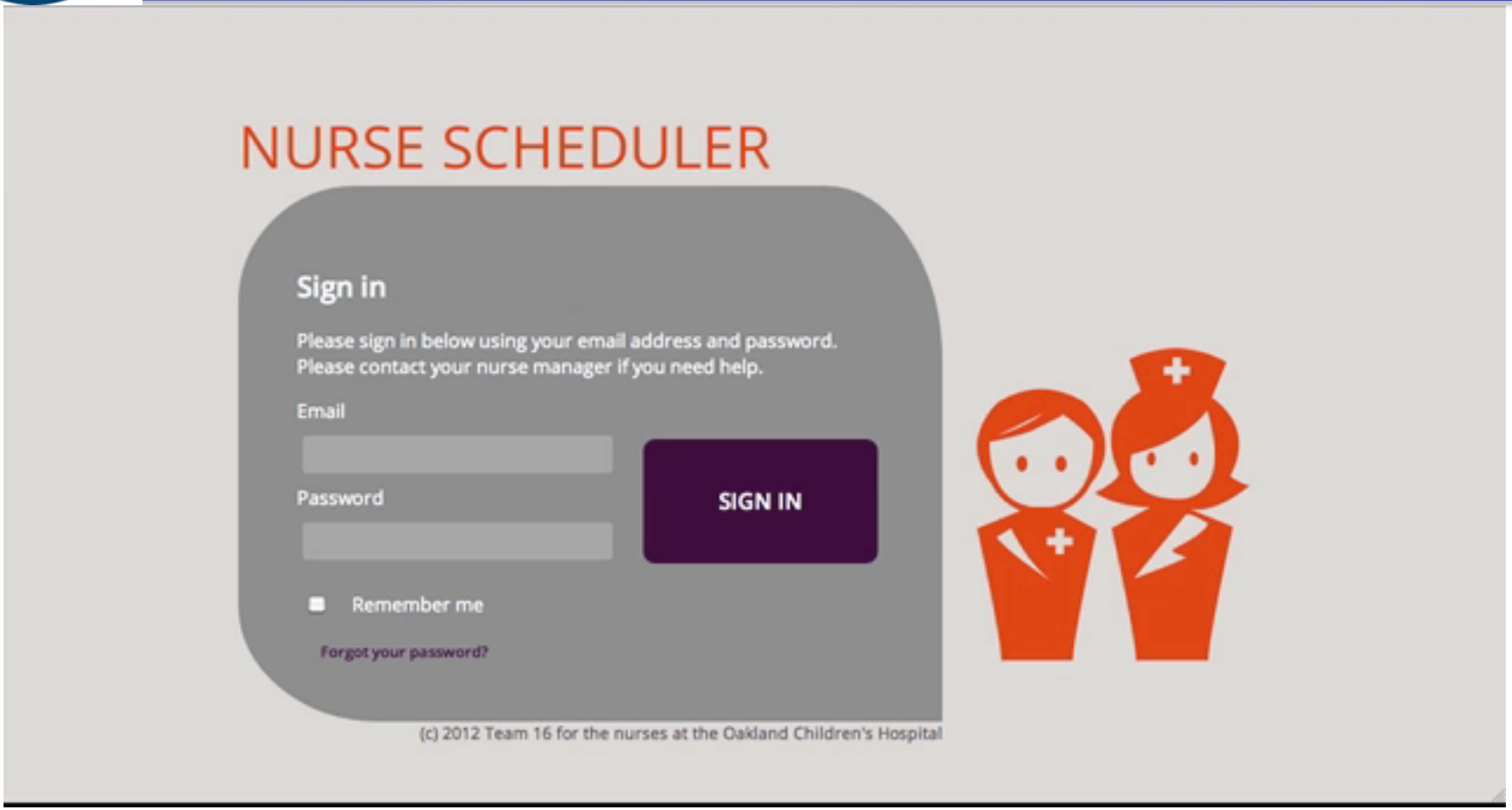
Revamping Berkeley Course

- Do project in "2-pizza teams"
- Recruit non-technical customer from non-profit organizations
  - Can't afford IT staff, or even to buy software
  - Grateful for any help

Example Non-Profit Projects

- Humane Society Pet Matchmaker
- Student Dormitory Package Notifier
- Minority VC firm Customer Relationship Manager (tracks startup proposals)
- Children's Hospital Nurse Vacation Scheduler (see video)
- Bonus: CS does community outreach
- Bonus: Led to new student organization
  - Blueprint, Technology for Non-Profits (http://bptech.berkeley.edu/)

Nurse Scheduler Project



Picking a Platform & Methodology

- Platform to motivate students
  - Smart Phone or Cloud Computing?
- SW Methodologies
  - Plan-and-Document or Agile?
- Pick combo with the best tools!
  - Save time given only 4-5 weeks
  - Easier for student to follow advice
  - Can grade process vs. final project

Picking a Platform & Methodology

- By large margin, best tools in Agile SW development for Cloud Computing Apps
  - "Software as a Service" or SaaS
- We're using Ruby on Rails
  - Ruby programming language
  - Rails programming framework
- Bonus: see lifelong learning of new tools is essence of SW Engineering

Agile Manifesto, 2001

"We are uncovering better ways of developing SW by doing it and helping others do it. Through this work we have come to value

- Individuals and interactions over processes & tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

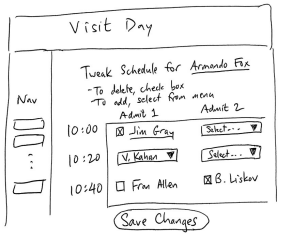
Agile lifecycle

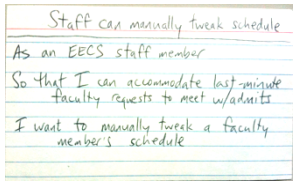
- Embraces change as a fact of life: continuous improvement vs. phases
- Developers continuously refine working but incomplete prototype until customers happy, with customer feedback on each Iteration (every ≈1 to 2 weeks)
- Agile: Test-Driven Development (TDD) to reduce mistakes, User Stories to validate customer requirements, Velocity (average no. user stories/ iteration) to measure progress

"Extreme Programming" (XP) version of Agile

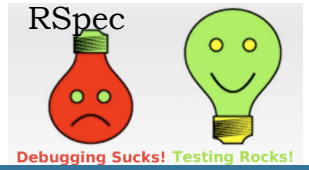
- If short iterations are good, make them as short as possible (weeks vs. years) => N iterations/project
- If simplicity is good, always do the simplest thing that could possibly work => Fewer lines of code
- If testing is good, test all the time; Write the test before you write the code to test => Serious testing
- If code reviews are good, review code continuously, by programming in pairs, taking turns looking over each other's shoulders => Peer learning
- Each helps classroom problem => Perfect for classroom

2-week Agile/XP Iteration





Cucumber



PIVOTAL TRACKER

heroku

Talk to customer

Lo-fi UI mockup

User stories & scenarios

Legacy Code

Design patterns

Behavior-driven Design / user stories

Test-first dev. (unit/funct.)

Measure Velocity

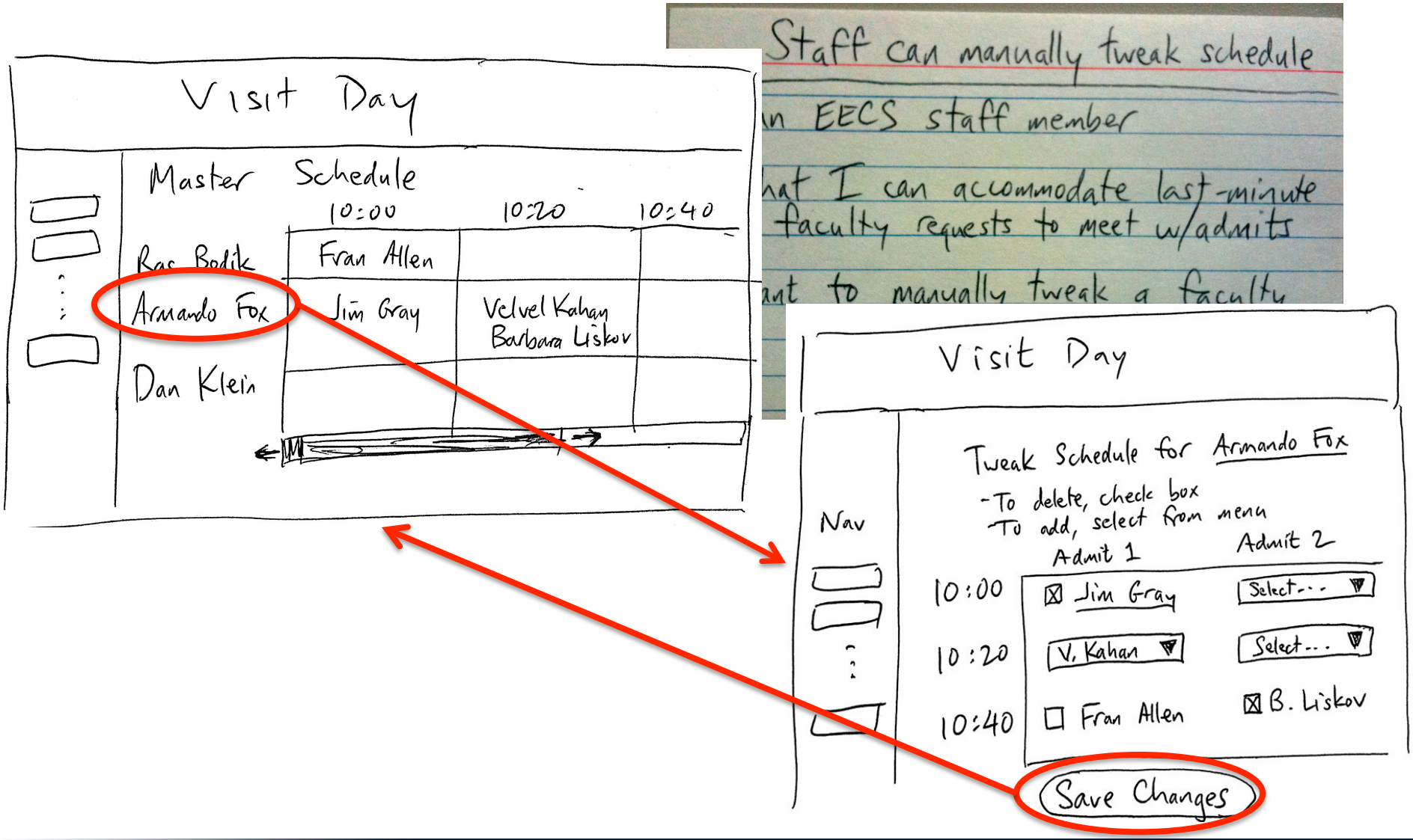
Deploy

Interview Nurse Customers

Methodologies …become Tools

• Software arch., design patterns, coding practices	• Ruby & Rails
• Test-first development, unit testing	• RSpec
• Behavior-driven design, integration testing	• Cucumber
• Agile, iteration-based project management	• Pivotal Tracker
• Version management & collaboration skills	• Git & Github
• SaaS technologies, deployment & operations	• Cloud computing: EC2, Heroku

Example: Behavior-driven Design from Lo-fi Mockup



Reaching agreement with customer via User Stories

```
Feature: staff can add admit to meeting with open slot
  As an EECS staff member
  So that I can accommodate last-minute requests
  I want to manually tweak a faculty member's schedule

Scenario: add an admit to a meeting with an open slot
  Given "Velvel Kahan" is available at 10:20
  When I select "Velvel Kahan" from the menu for the 10:20
       meeting with "Armando Fox"
  And I press "Save Changes"
  Then I should be on the master meetings page
  And I should see "Velvel Kahan added to 10:20AM meeting."
  And "Armando Fox" should have a meeting with "Velvel
       Kahan" at 10:20

Scenario: remove admit from meeting
```

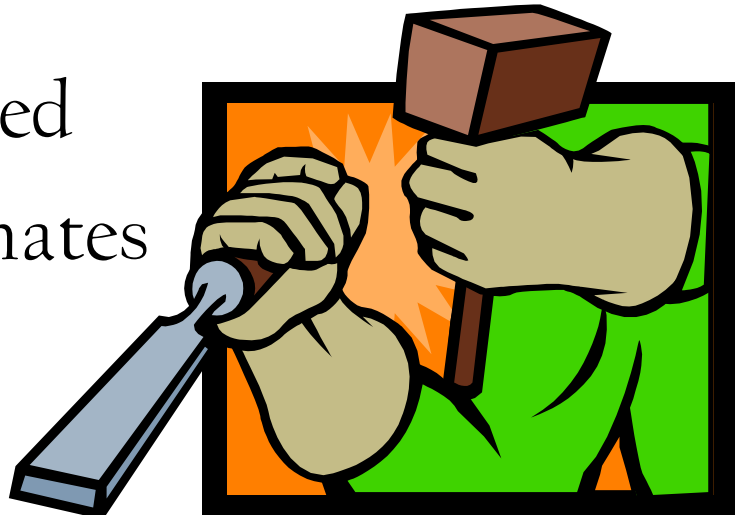
...

From user stories to acceptance tests

- Runs "natural language" user stories as integration tests
- Each *scenario* describes one user story
  - *Given* steps: setup preconditions
  - *When* steps: take actions, using built-in browser simulator
  - *Then* steps: assertions to check post-conditions
- *Step definitions* match story steps to code via regexes
- Quantify correctness and coverage

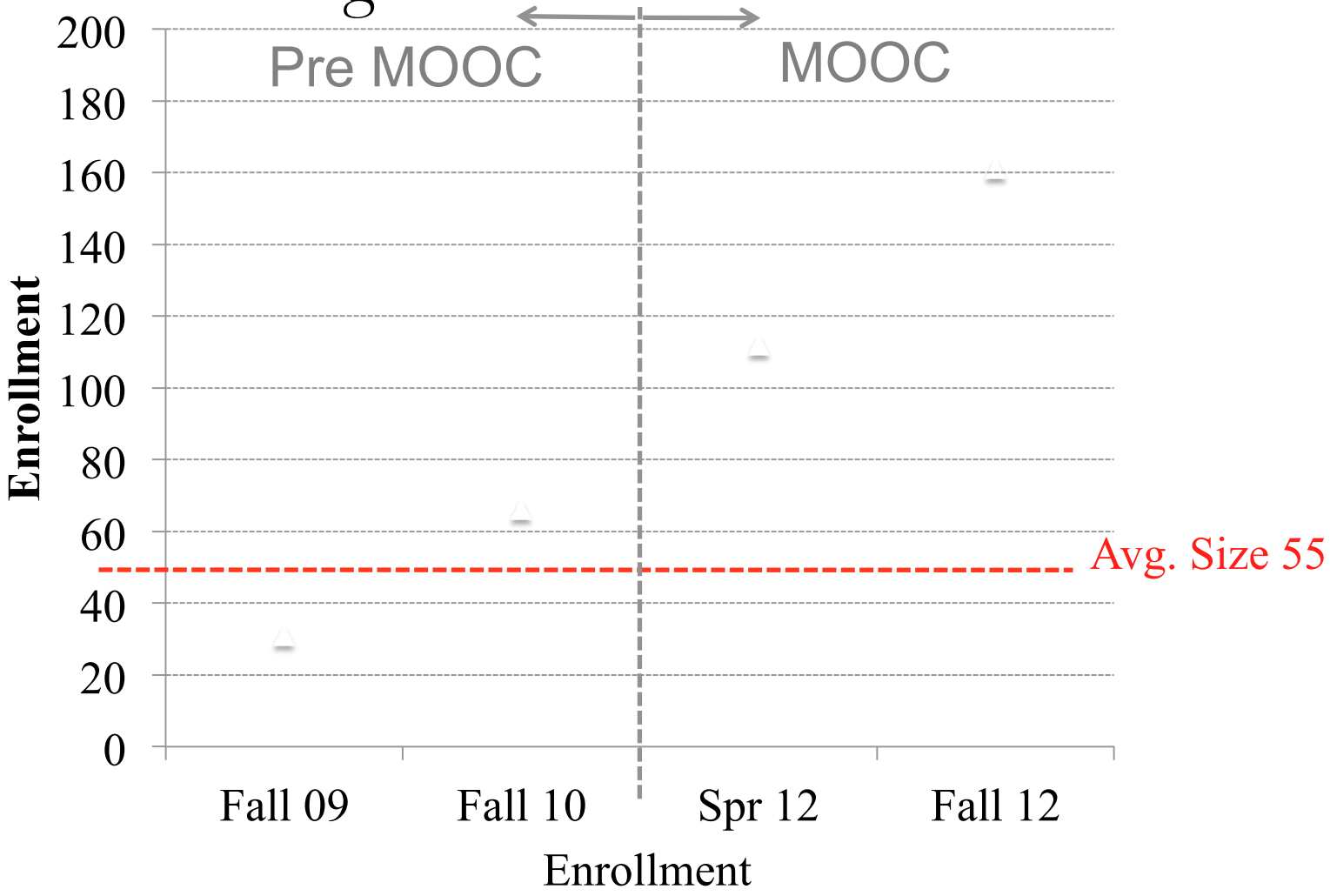
Methodologies ➔ Tools

- Students more easily follow advice
- Instructors more easily grade
- Per-iteration progress quantified
- Students get feedback on estimates
- All these tools free, some hosted in cloud



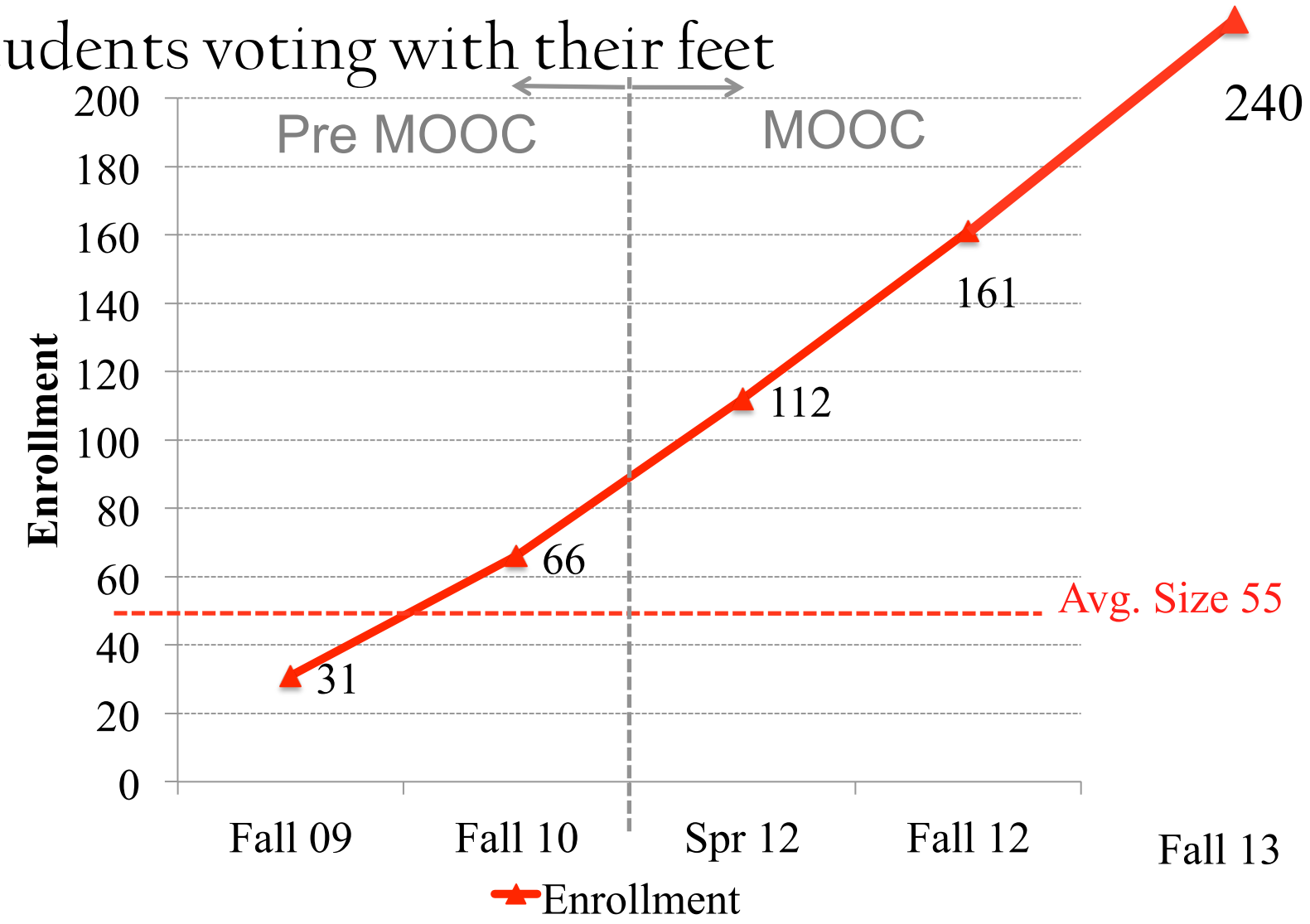
Evaluating Agile Approach

- Students voting with their feet



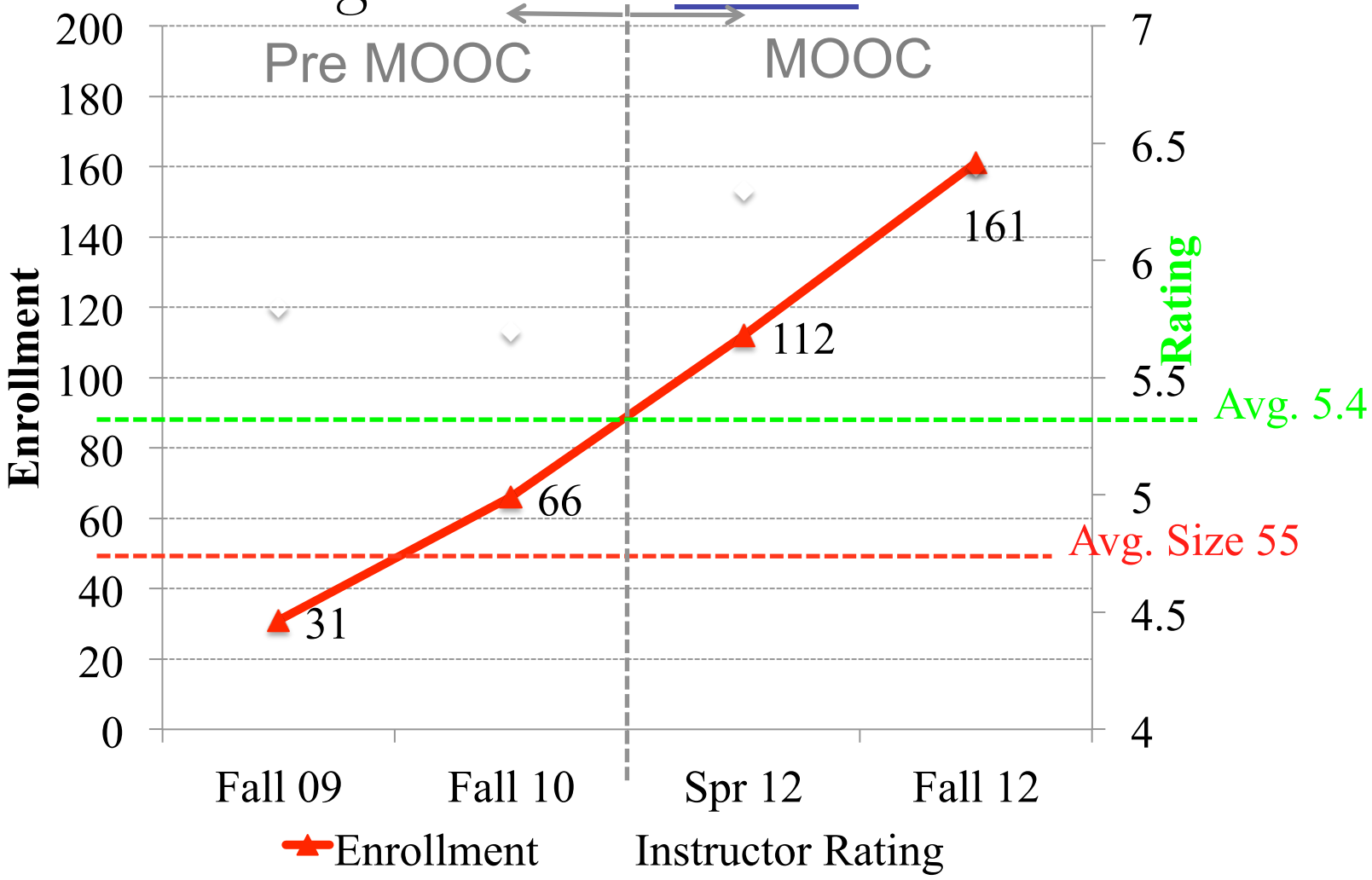
Evaluating Agile Approach

- Students voting with their feet



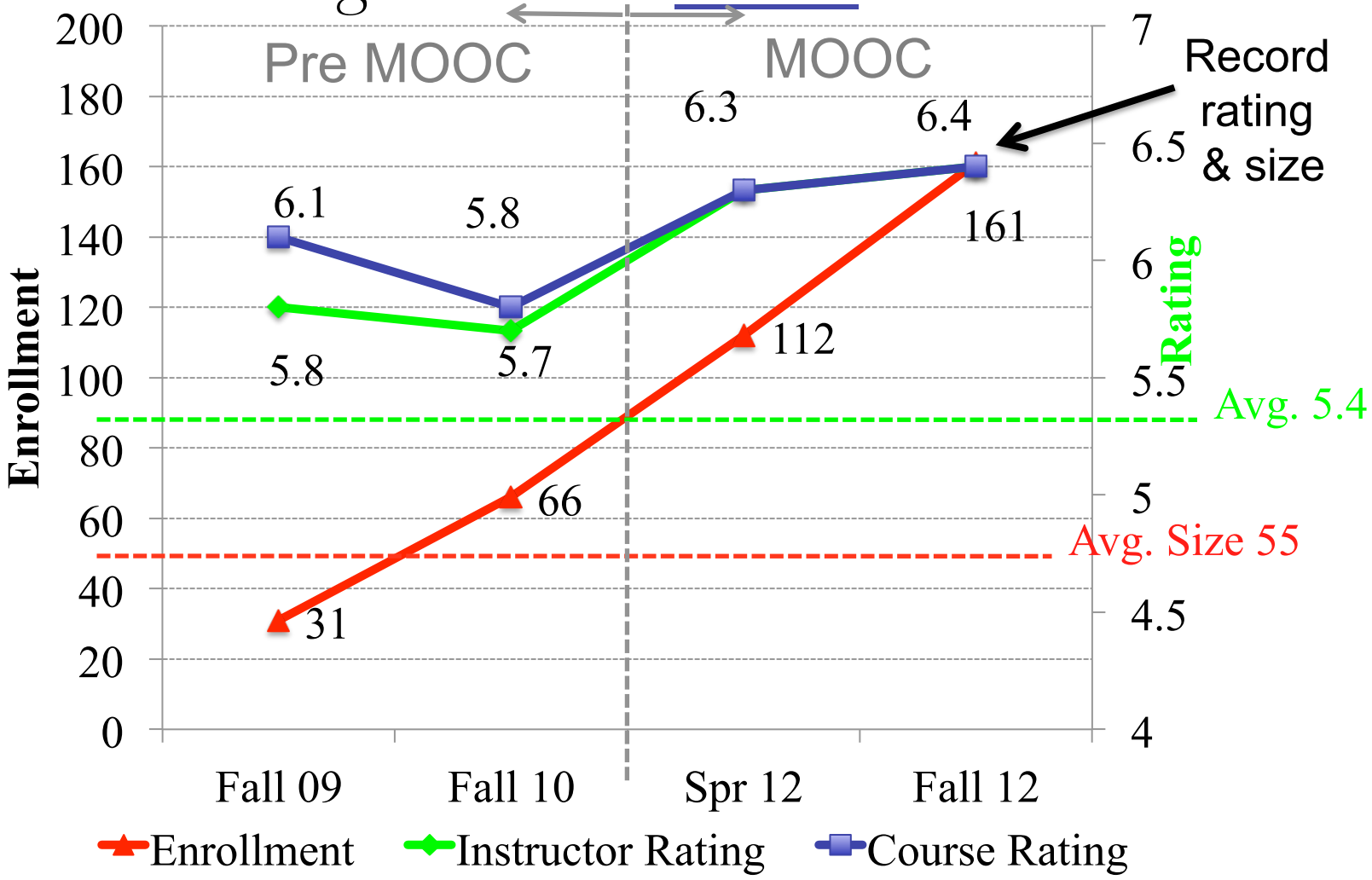
Evaluating Agile Approach

- Students voting with their hands



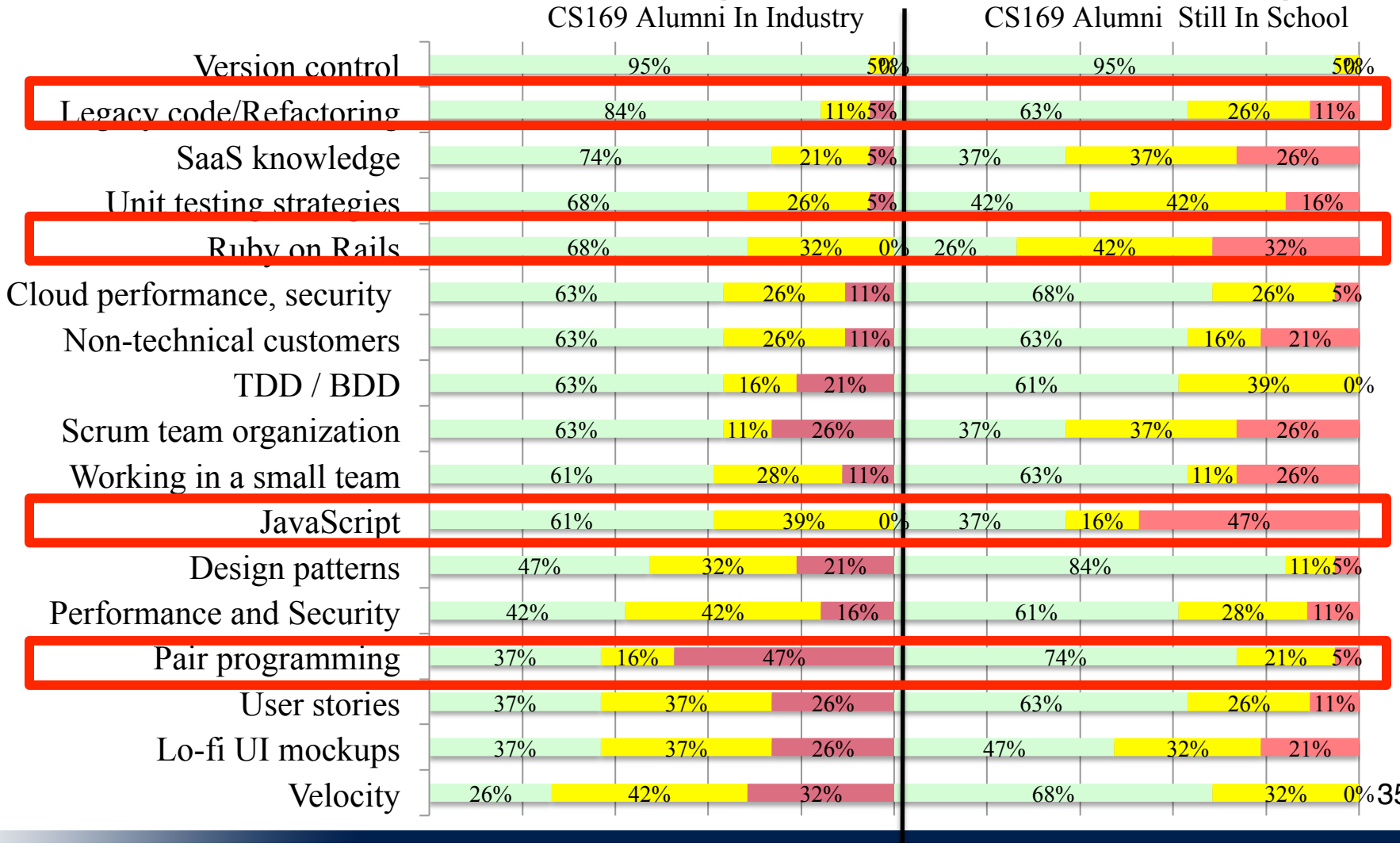
Evaluating Agile Approach

- Students voting with their hands



Evaluating Agile Approach

- Topic useful in my work? **Agree** – **Neutral** - **Disagree**



Evaluating Agile Approach

- (Anecdotal) Industrial Feedback

I'd be far more likely to prefer graduates of this program than any other I've seen.
— Brad Green, Engineering Manager, Google Inc.

A number of software engineers at C3 Energy consistently report that this . . . course enabled them to rapidly attain proficiency in SaaS development. I recommend this . . . course to anyone who wants to develop or improve their SaaS programming skills.
—Thomas Siebel, CEO C3 Energy, founder & CEO Siebel Systems

- Non-Technical Customer Feedback
  - 48% customers tried to hire students
  - 92% customers "happy" or "thrilled" (see video)

Curriculum Committee Agrees

- "students learn best … by participating in a project … Utilizing project teams, projects can be sufficiently challenging to require the use of effective software engineering techniques…"
- "students better learn to apply software engineering approaches through an iterative approach …[they] assess their work, then apply the knowledge gained through assessment to another development cycle"

Joint Task Force on Computing Curricula, "Computer Science Curricula 2013, Ironman Draft (version 1.0)," ACM/IEEE CS, Feb. 2013

Outline

- Myths about Software Engineering Education
- Reinventing SW Engineering with Agile+SaaS
- Myths about MOOCs (Massive Open Online Course)
- Experience with SPOCs (Small Private Online Course)
- Conclusion: SPOC+Ebook == 21st Century Textbook (curricular technology transfer)

MOOC Surprise

- 6 months after decide to write Ebook, recruited for Massive Open Online Course
  - 1st Berkeley MOOC
  - 1st or 2nd MOOC from Coursera
  - 10,000 earned certificates in 2012

- Ebook/textbook and MOOC developed together
  - Each Ebook section maps to 1:1 to MOOC video segment
  - Recorded MOOC 3 times => 3 editions of *Engineering Software as a Service*: Alpha, Beta, 2nd Beta

What Enabled MOOCs?

1980 1. Reuse Prof investment in course prep/ lecture, so lecture videos are "free"
   - We recorded live lectures vs. studio recording

2006 2. Free, scaled up video distribution: YouTube

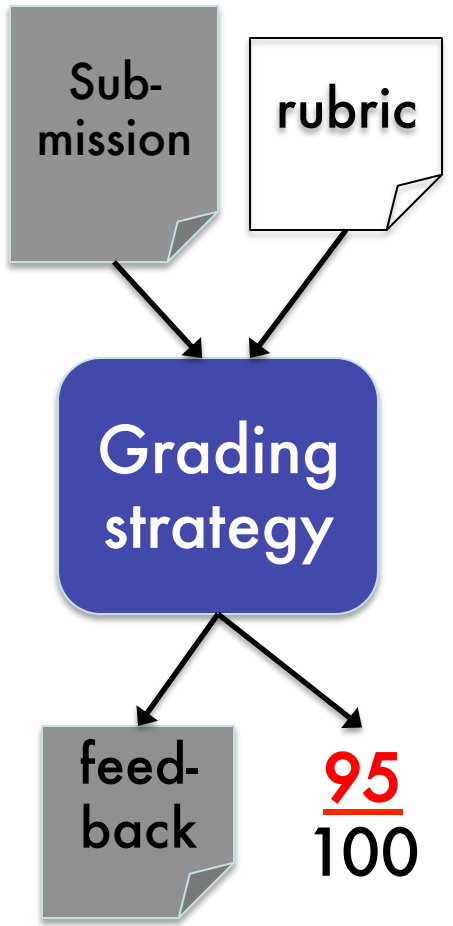
2006 3. 10 minute segments >> 60 minute lectures
   - Discovered by Khan Academy

2008 4. Scaled up question answer: By students & TAs in online forums (≈Stack Overflow)

2006, 2010 5. Scaled up grading of exams & programming: Cloud computing + Rails testing tools

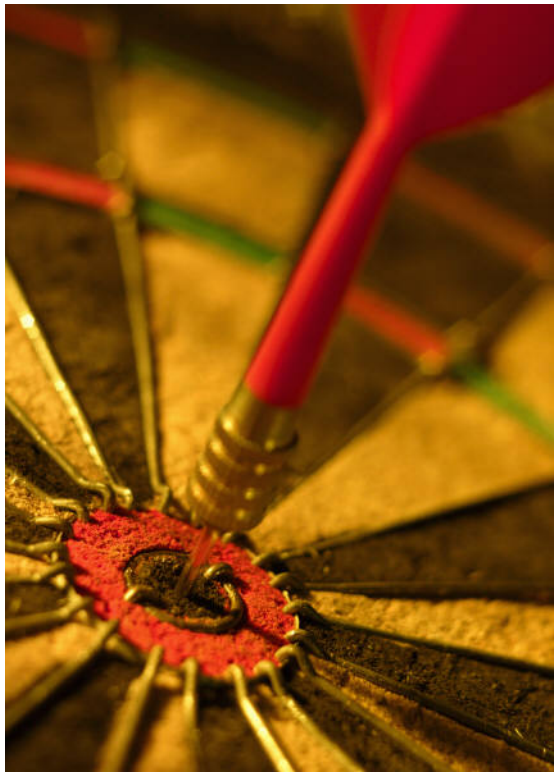
Autograding Strategies

Assignment type	Grading strategy
Write code	· RSpec (correctness) ·[now] `reek/flay` (code style) ·[now] `CodeClimate.org` (metrics)
Write test cases (unit, functional, or user stories)	· Mutation testing (Amman & Offutt): app with inserted bugs should cause some tests to fail
Enhance legacy SaaS app (deploy on Heroku)	· Remote (cloud-based) integration test using `Mechanize` · C0, happy path, sad paths coverage
Interactive short-answer/multiple-choice	· Our tools emit both printed & online-format (XML) quizzes · [soon] open-ended short-essay



Software Distribution?

- Virtual Machines preloaded with everything students need
  - With correct versions to match Ebook, lecture
  - We used VirtualBox
- Amazingly, few problems with 1000s of students (forum resolved)
  - Except fast enough computer (some netbooks too slow?)
  - And portal to download VMs (thanks to AWS, Google, Microsoft)



MOOC Myth 1: Threat to US undergrad programs

- 80% outside US (10,000 certificates in 113 countries)
  - US (20%), Spain (10%), India (7%), Russia (6%), UK (5%), Brazil (4%), Canada (3%), Ukraine (3%), Germany (2%)

Highest Degree	
< High school degree	1%
High school degree	8%
Some college, no degree	11%
Associate degree	3%
Baccalaureate	32%
Prof. degree (JD, MD, ...)	11%
Grad degree (MS, PhD, ...)	35%

Primary Occupation	
High school student	1%
Undergraduate student	8%
Graduate student	5%
Raising family at home	1%
Full time job	70%
Part time job	6%
Unemployed	8%

- Reality: Threatens continuing education programs

MOOC Myth 2: Course suffers if replaced by MOOC

- "Autograding can't replace instructor help"
  - Can it level-up student confidence & raise productivity of instructor interactions?
- "Online can't replace classroom discussion"
  - What foundational skills can online strengthen?
- "Online interaction can't replace face to face"
  - How & why does perceived community in online courses improve student engagement & retention?*
- MOOC minus "unMOOCable" parts still has value (e.g. campus course's design project not in MOOC)

* J.C. Richardson & K. Swan, *Examining Social Presence in Online Courses in Relation to Students' Perceived Learning and Satisfaction,* J. Async. Learning Networks 7, 2003

MOOC Myth 3: Will weaken on-campus pedagogy

- Berkeley: MOOC improved evaluations
- Enough students to use inferential statistics techniques (SAT exams)
  - *Exploratory factor analysis*: test comparable concepts, can vary exams
  - *Item response theory*: which questions more difficult for good students
  - *A/B testing*: which approaches lead to better learning outcomes

- Reality: can help to improve on-campus pedagogy

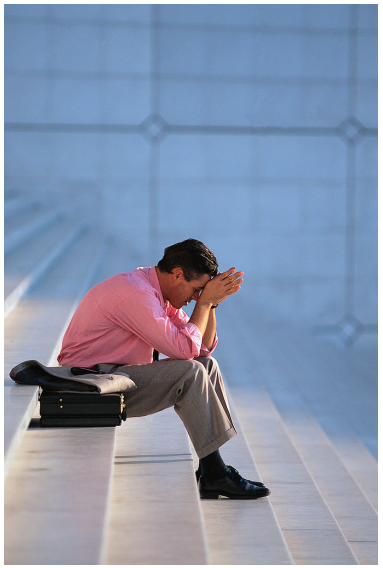
MOOC Myth 4: Distract faculty & hurt productivity

- Berkeley: 4X students in SW Eng course
- SJSU tried EE MOOC from MIT
  - MOOC homeworks, lectures
  - Same exams as prior SJSU course
  - Average 5% higher 1st exam
  - Average 10% higher 2nd exam
  - 91% got C or better (59% before)
  - More students finish course

- Reality: Can improve faculty productivity

MOOC Myth 5: Profs now TAs for homogenized courses

- Small Private Online Course (SPOC) allows faculty to create a la carte course
- Automatically graded assignments?
- MOOC Forum to answer questions?
- Use videos
  - To help prepare lectures?
  - Or as portion of lectures? (e.g., pieces unsure about?)
  - Or as substitute when instructor travels?
  - Or to "flip classroom" if prefer tutoring to lecturing?
- Reality: SPOCs aid faculty, improve their morale



Outline

- Myths about Software Engineering Education
- Reinventing SW Engineering with Agile+SaaS
- Myths about MOOCs (Massive Open Online Course)
- Experience with SPOCs (Small Private Online Course)
- Conclusion: SPOC+Ebook == 21st Century Textbook (curricular technology transfer)

The SPOC Experience

- Technology developed can scale up to MOOCs, scale down to SPOCs
  - Improve quality of lesser-known programs?
- Tried 5 SW Eng SPOCs Spring 2013
  - Binghamton University, NY
  - Hawaii Pacific University
  - Tsinghua University, Beijing
  - U. of Colorado, Colorado Springs
  - U. of North Carolina, Charlotte

SPOC Selection Spectrum

	Binghamton	*UC Berkeley*	*U. Colorado Colorado Springs*	*Tsinghua*	*UNC Charlotte*	*Hawaii Pacific*
Engineering SaaS Textbook	✔	✔	✔	✔	✔	✔
Instructor Reviews Video	✔	✔	✔	✔	✔	✔
Reuse Assignments	✔	✔	✔	✔	✔	✔
Autograde Assignments		✔	✔	✔	✔	✔
Reuse Exams	✔		✔	✔	✔	
Have Local SPOC Forum		✔		✔	✔	✔
Reuse Lecture Slides	✔	✔	✔			
Students Video Optional		✔	✔			
Show videos in some lectures			✔			
Flip Classroom (video required)				✔	✔	
Online Course (video required)						✔
Autograded Exams						

SPOC Good & Bad

- Auto-graders took grading burden off staff & emphasized TDD
- Video lectures dense, efficient (can rewind)
- Students excited about latest tech (Rails, Agile)
- Students impressed with "world-class" instruction
- Students got jobs/started companies based on class material

- Given MOOC, answers available on Internet
- Auto graders checked for correct "output," but did not check code style (fix this semester)
- Some student computers too slow to run VM
- Some didn't know Linux
- Some reduced work if hit logistical problems

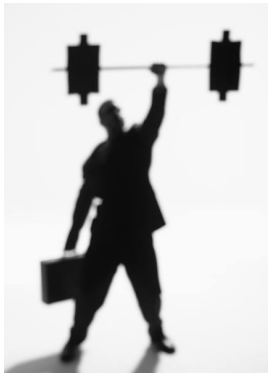
SPOCs Next Time

- Try participating in global MOOC Forum?
  - Talk to students other schools about common problems
  - Broaden SWE perspective
  - Access to "World TAs" to help with technology (Ruby, Rails, tools)
- All 5 want to do again
  - + Want to add more universities

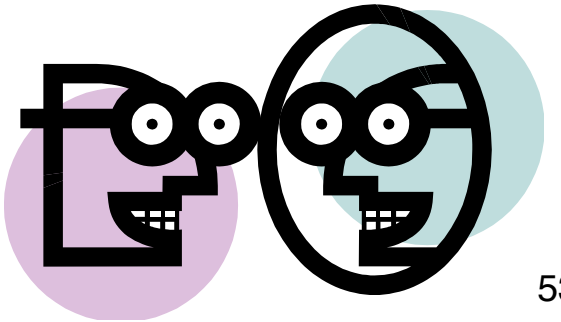
Conclusions: Campus SW Engineering works now

- SW Eng Challenges
  - SWE unnatural, limited time, novice teachers, many methodologies, poor textbooks, no tools, & industry criticizes

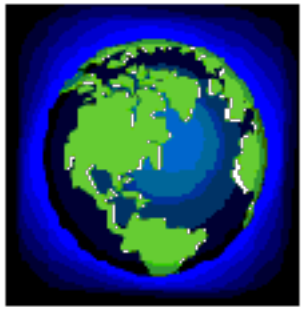
- SaaS & Agile revolutionizing SW industry + making SW Eng *easier* to teach
  - SaaS tools => multiple iterations => better code => learn, use & appreciate SWE concepts



- Faculty, students, & industry now embrace revised SW Eng course



Conclusions: MOOC + Ebook = 21st c. curriculum tech transfer?



- MOOCs increase *size* and *scope* of course & *repetitions of* course
  - Auto graders, videos, forums, World TAs

- Need Ebook: MOOC alone too challenging for students (and faculty)
- SPOCs+Ebooks expand *number of* classrooms as well as empower faculty and improve their productivity

- 21st Century textbook: SPOC/Ebook hybrid?

To Learn More: beta.saasbook.info/about

- Estler, H.-C., et al., "Agile vs. Structured Distributed Software Development: A Case Study." *Proc. 7th Int'l Conf. Global Software Eng.*, IEEE 2012, pp. 11–20.
- Fox, A. "From MOOCs to SPOCs." *CACM*, to appear.
- Fox, A., & Patterson, D. *Engineering Software as a Service*, 2nd Beta Edition, Strawberry Canyon, 2013.
- Fox, A., & Patterson, D. "Is the New Software Engineering Curriculum Agile?" *IEEE Software*, 30:5, Sept/Oct 2013.
- Fox, A., & Patterson, D. "Crossing the software education chasm." *CACM*, 55:5, May 2012.

Thanks!

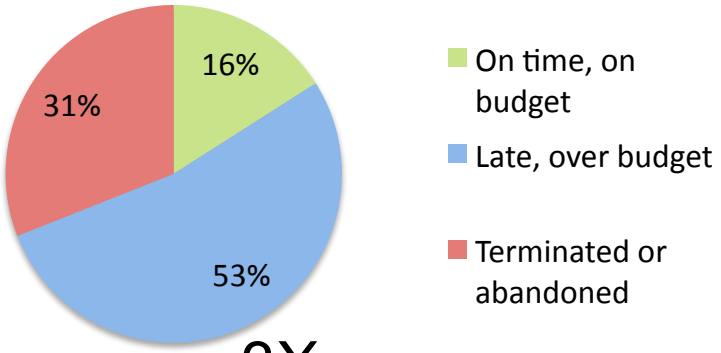
More info (papers,...): saasbook.info

Acknowledgments: David Patterson, staff of UC Berkeley CS 169, support staff for EdX CS 169.1x/169.2x

Backup Slides

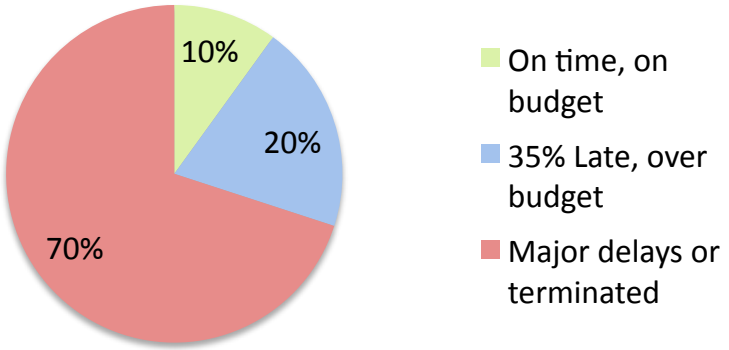
How Well do Plan-and-Document Processes Work?

Software Projects (Johnson 1995)

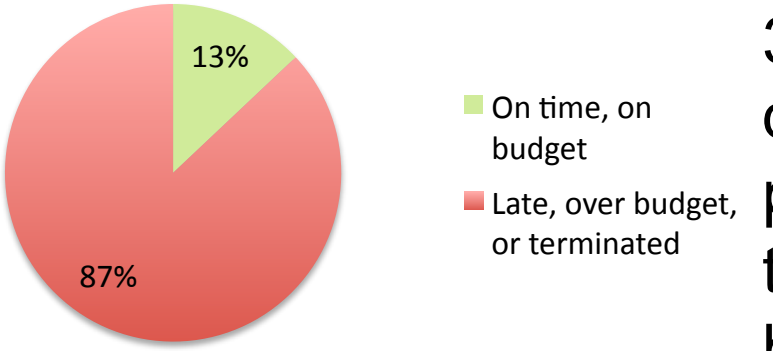


3X

Software Projects (Jones 2004)



Software Projects (Taylor 2000)



3/~500 new development projects on time and budget

J. Johnson. *The CHAOS report*. Technical report, The Standish Group, Boston, Massachusetts, 1995.

A. Taylor. IT projects sink or swim. *BCS Review,* Jan. 2000.

C. Jones. Software project management practices: Failure versus success. *CrossTalk: The Journal of Defense Software Engineering*, pages 5–9, Oct. 2004.

(Figure 1.6, *Engineering Long Lasting Software* by Armando Fox and David Patterson, 2nd Beta edition, 2013.)

Want to do MOOC yourself?

- Having a Rerun Plan is Better than Being Perfect
  - Needed feedback from MOOC students before we could improve it ourselves
- Consider Delegating
  - MOOC alumni volunteer as "World TAs"
- Dry Run the Technology
  - With 1000s of students, must be perfect
- Divide to Conquer
  - Divided 12 weeks lecture into two 6-week MOOCs

Ruby on Rails Too Slow?

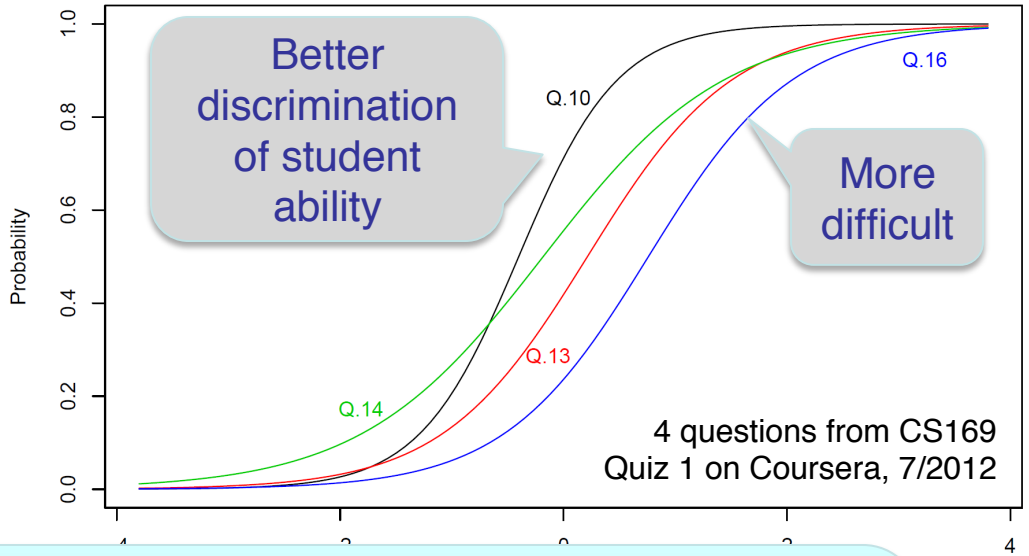
- Computer cost-performance improvements
  - 1000X since Java announced in 1995
  - 1,000,000X since C++ announced in 1979
  - Spend on programmer productivity (in classroom)
- In the Cloud, horizontal scalability can trump single-node performance
  - Can teach (and test) what makes an app scalable
  - Not covered elsewhere in curriculum

Rails Productivity?

- Compared to Java and its frameworks, Rails programmers found 3X – 5X reductions in number of lines of code
  - Feng, J. & T. Sedano. "Comparing Extreme Programming and Waterfall Project Results" *Conference on Software Engineering Education and Training* (2011).
  - Stella, L., S. Jarzabek, & B. Wadhwa, "A comparative study of maintainability of web applications on J2EE, .NET and Ruby on Rails," *WSE 2008. 10th International Symposium on Web Site Evolution*, Oct. 2008.

Scale can accelerate education innovation

- **Item response theory** Predicts probability that a student of a given ability will answer a given question correctly
- Do
- Ca

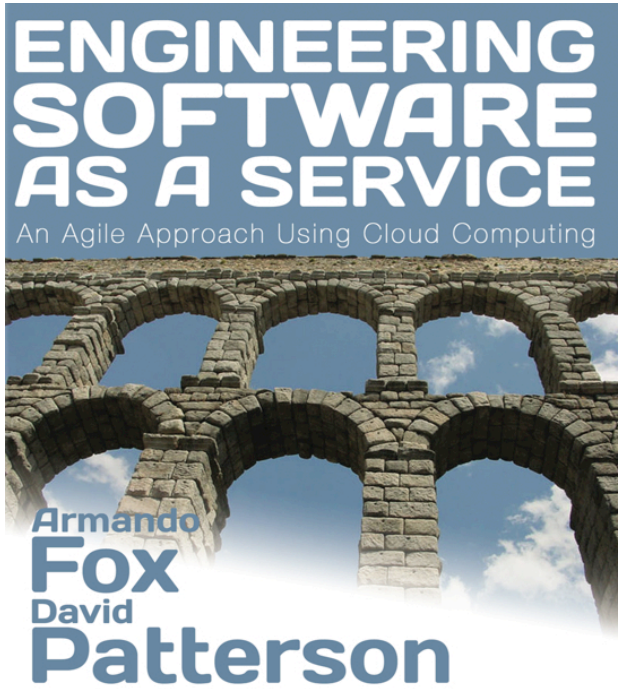


Large # of students reduces standard error of question difficulty & discrimination model by 3x-10x.

* Frederic M. Lord, *Statistical Theories of Mental Test Scores* (1968) and *Applications of Item Response Theory to Practical Testing Problems* (1980)

Ebook Myth: Manufacturing Cost 0$, so price should be 0$

- Authors get 15-30% of avg. selling price (85% of list price)
- Not counting ≈ 2000 hours of writing time, we invested >$15k in Ebook development
  - Cover, index, artwork, Ebook element design, format conversion, marketing, advertising, …
- However, if self publish, can cut out middleman (our Ebook is $10)



Reviews of *Engineering Software as a Service*

- Amazon Rating (Sept 2013): 4.7 / 5 stars

- *"Great review of modern SAAS and agile software engineering techniques."*
- *"Great intro to agile software development."*
- *"Well worth the cost for online version"*
- *"Classroom focused"*
- *"Originally bought for EdX course - but it is a great book for learning what modern web developers should know"*
- *"Useful, up to date, fun to read"*
- *"Fun to read, thoroughly researched"*
- *"Wow"*

Full Quote Team Projects

- *In general, students learn best at the application level much of the material defined in the [software engineering knowledge area] by participating in a project. Such projects should re-quire students to work on a team to develop a software system through as much of its lifecycle as is possible. Much of soft-ware engineering is devoted to effective communication among team members and stakeholders. Utilizing project teams, projects can be sufficiently challenging to require the use of effective software engineering techniques and that students develop and practice their communication skills. While organizing and running effective projects within the academic framework can be challenging, the best way to learn to apply software engineering theory and knowledge is in the practical environment of a project.*

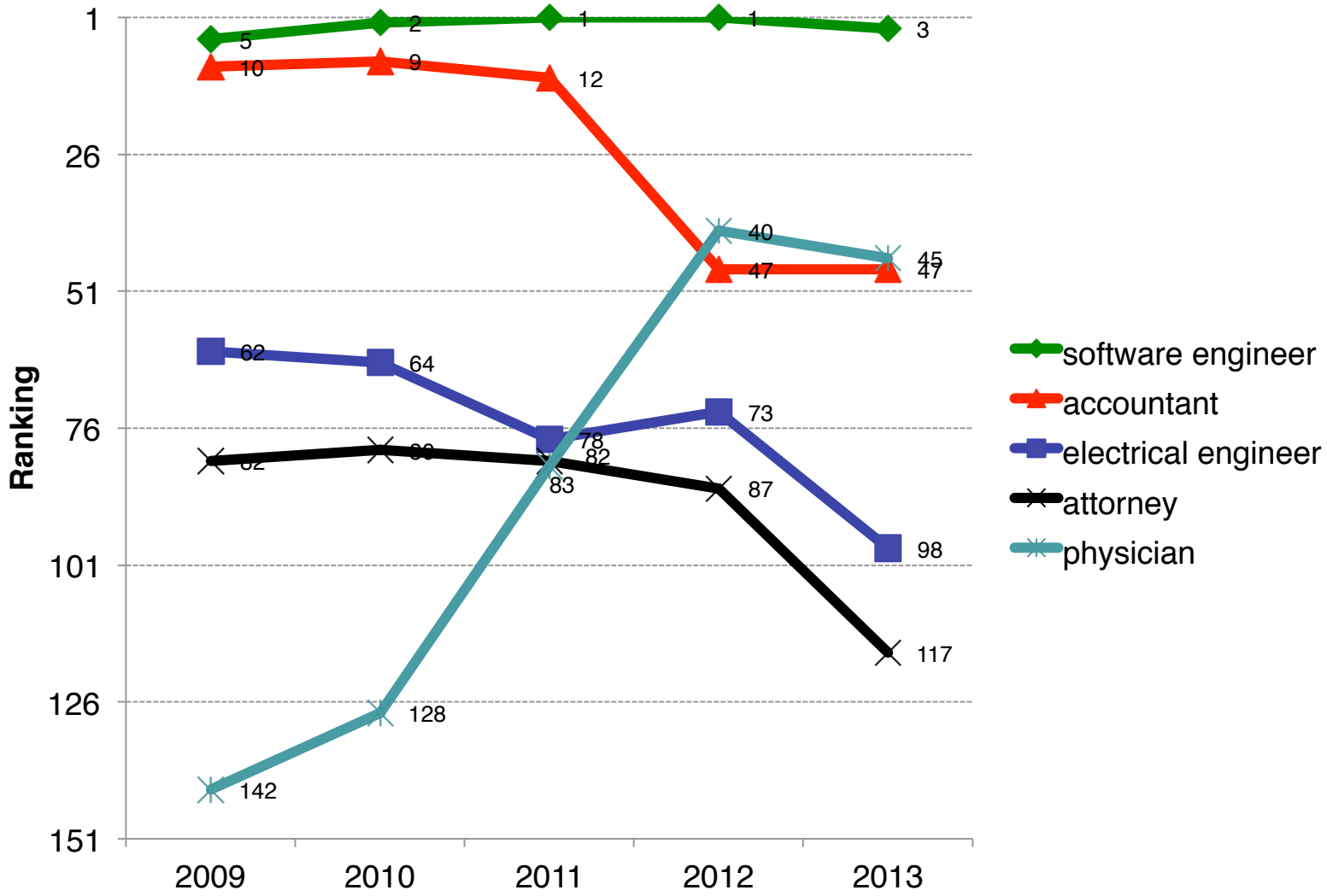
Joint Task Force on Computing Curricula, "Computer Science Curricula 2013, Ironman Draft (version 1.0)," ACM/IEEE CS, Feb. 2013

Full Quote Agile

- *. . . there is increasing evidence that students better learn to apply software engineering approaches through an iterative approach, where students have the opportunity to work through a development cycle, assess their work, then apply the knowledge gained through their assessment to another development cycle. Agile and iterative lifecycle models inherently afford such opportunities.*

Joint Task Force on Computing Curricula, "Computer Science Curricula 2013, Ironman Draft (version 1.0)," ACM/IEEE CS, Feb. 2013

Ranked Jobs: 2009-2013



Source: careercast.com (income, job outlook, stress, work env.)