Azrieli
College of Engineering
Jerusalem

עזריאלי
מכללה אקדמית להנדסה
ירושלים

Lessons From Teaching a Software Engineering for Cloud Computing Course - From Class to MOOC to SPOC

Reuven Yagel

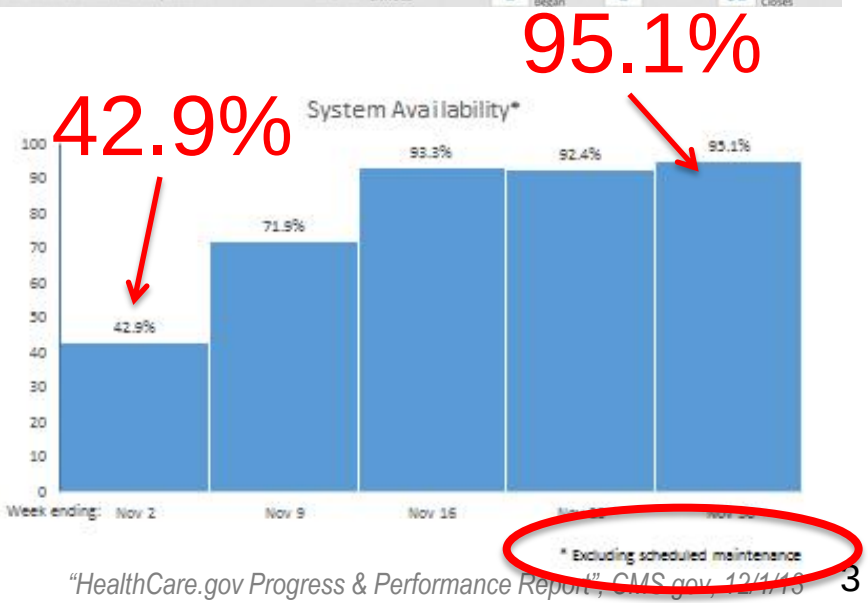
Azrieli - College of Engineering, Jerusalem

Based on: http://beta.saasbook.info/about/presentation

Using MOOCs to Reinvigorate Software Engineering Education

Armando Fox & David Patterson
University of California, Berkeley

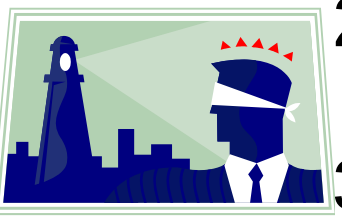
Poor President Obama…



"HealthCare.gov Progress & Performance Report", CMS.gov, 12/1/13

"Programming project" course ≠ software engineering…

1. Limited time to teach (in most CS degrees)
   - 1 semester @ 12 credit-hrs/week ≈ 4-5 weeks FTE



2. Most instructors aren't experts
   - Berkeley: 16 faculty in 20 years taught SWE!
3. Many methodologies; which to teach?
   - "Plan & document" or "Agile" family?

4. Popular texts are surveys vs. learn by doing
   - "*I hear & I forget, I see & I remember, I do & I understand*" (Confucius)
   - Yet 895-page survey, 32 chapters, 1.7★ (out of 5)

Leading textbook?

- Pressman, *Software Engineering,* 7th ed. (1st ed. 1982)
- Amazon Rating 1.7★ / 5, price $199 (ebook

• "*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• "*Horrible out-of-date Software Engineering book*"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• "*Buy only if you want a narrow view of plenty of outdated topics*"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*"

Result: poor preparation for software careers

- Common case 1: Students ignore lectures, build SW as they always have
  - "just a project course"
- Common case 2: Students learn & use methodologies that are *known to be likely to fail* for large projects
- Result: "hot potato" course
  - frustrating to instructors (get poor ratings)
  - boring to students
  - disappointing to industry

Outline

- State of Software Engineering Education
- Reinventing SW Engineering with Agile+SaaS
- Description & Evaluation of Revamped Course
- Curricular Tech Transfer: MOOCs & SPOCs
- Local Lessons

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 as team with non-technical customer
  3. Automated testing

How to redesign course to address these gaps?

Choosing a Methodology: 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
- Small team of developers continuously refines working but incomplete prototype until customers happy, with feedback on each Iteration (~1-2 weeks)
  - Behavior-Driven Design & User Stories to elicit & validate customer requirements
  - Test-Driven Development (TDD) to reduce mistakes
  - Velocity (weighted average of stories completed/iteration) to measure progress

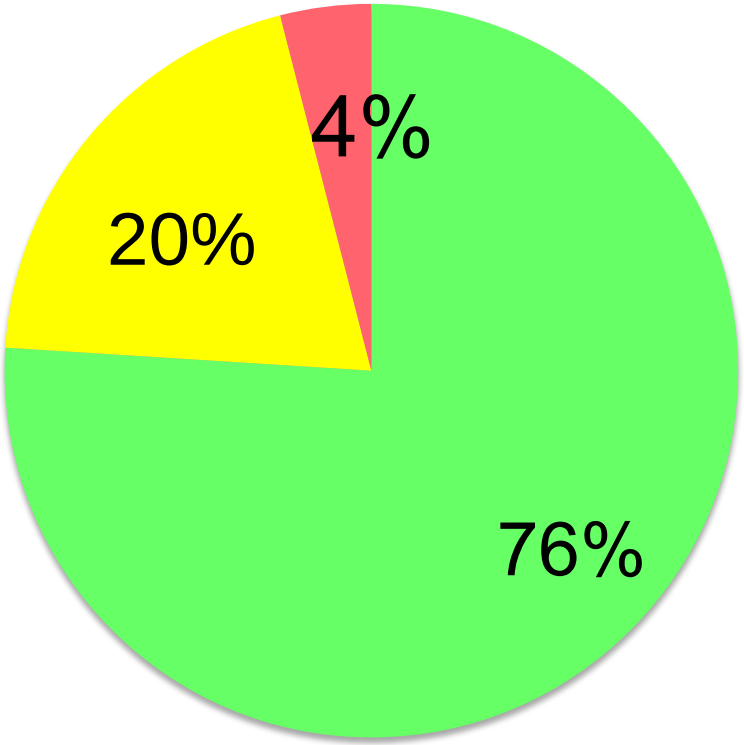
“Extreme Programming” (XP) version of Agile

- 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

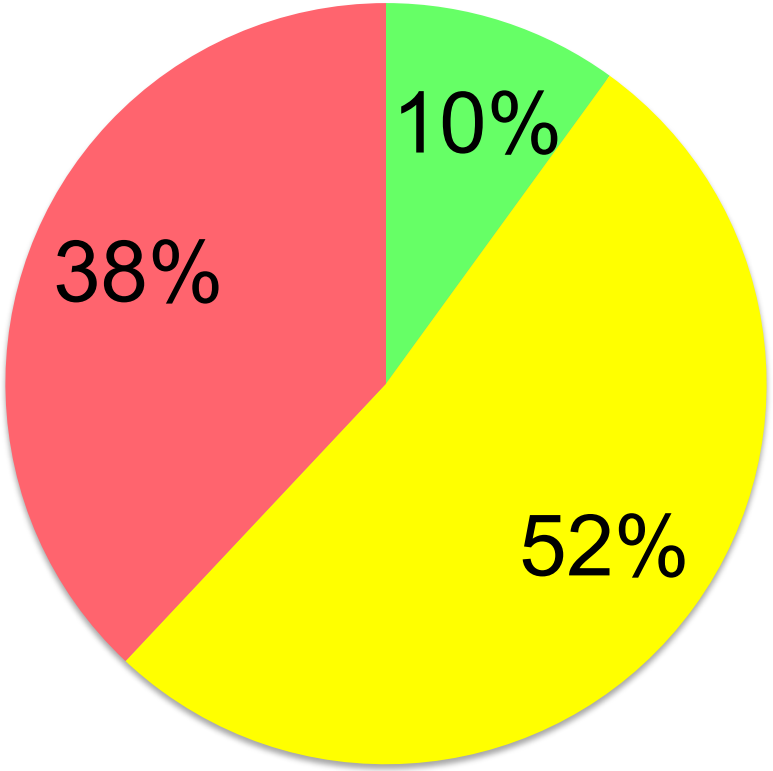
➔Perfect for classroom

Does Agile Work?

Small (Agile) projects, <$1M



Large (non-Agile) projects, >$10M

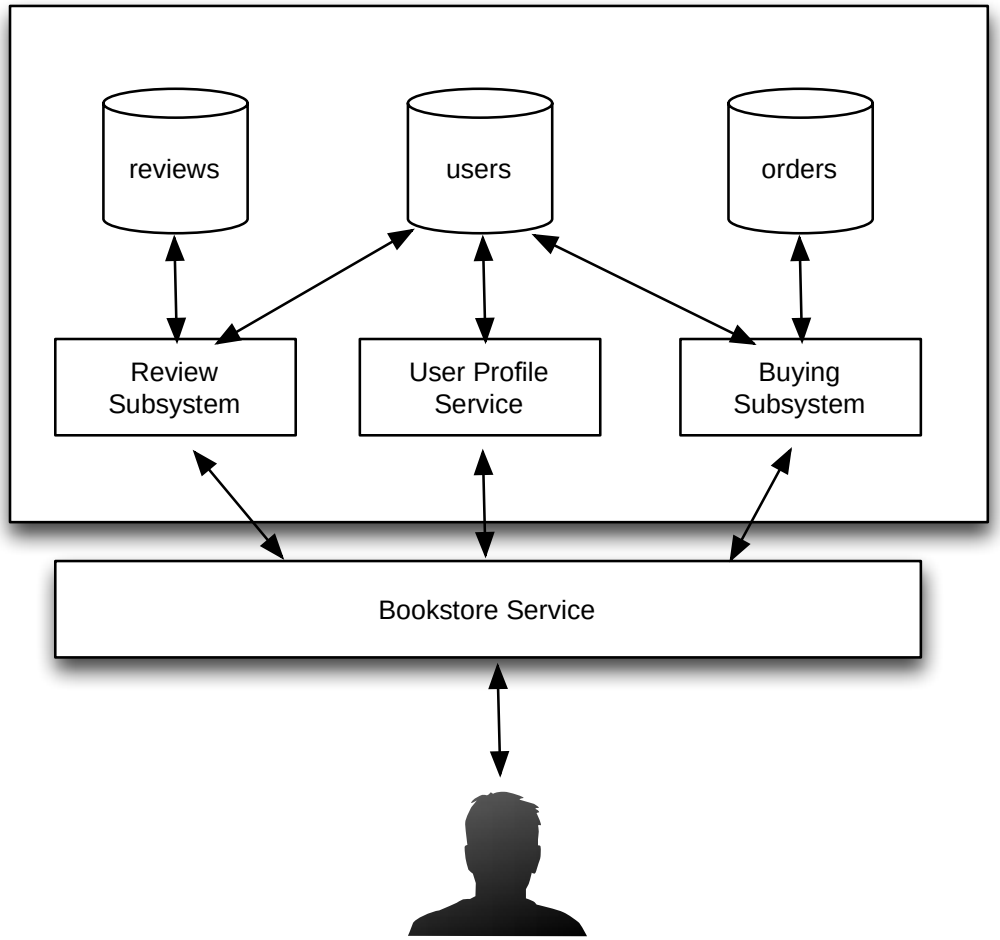


Agile is popular: 55%–80% of teams in 2012

CHAOS MANIFESTO 2013 Think Big, Act Small, www.standishgroup.com.
Based on the collection of project case information on 50,000 real-life IT environments and software projects. Surveying since 1985.

Is Agile only good for small projects/systems?

- Typical large system: Internal subsystems share data directly
  - eg, Reviews module accesses User Profiles
- All subsystems inside single API ("The Bookstore")



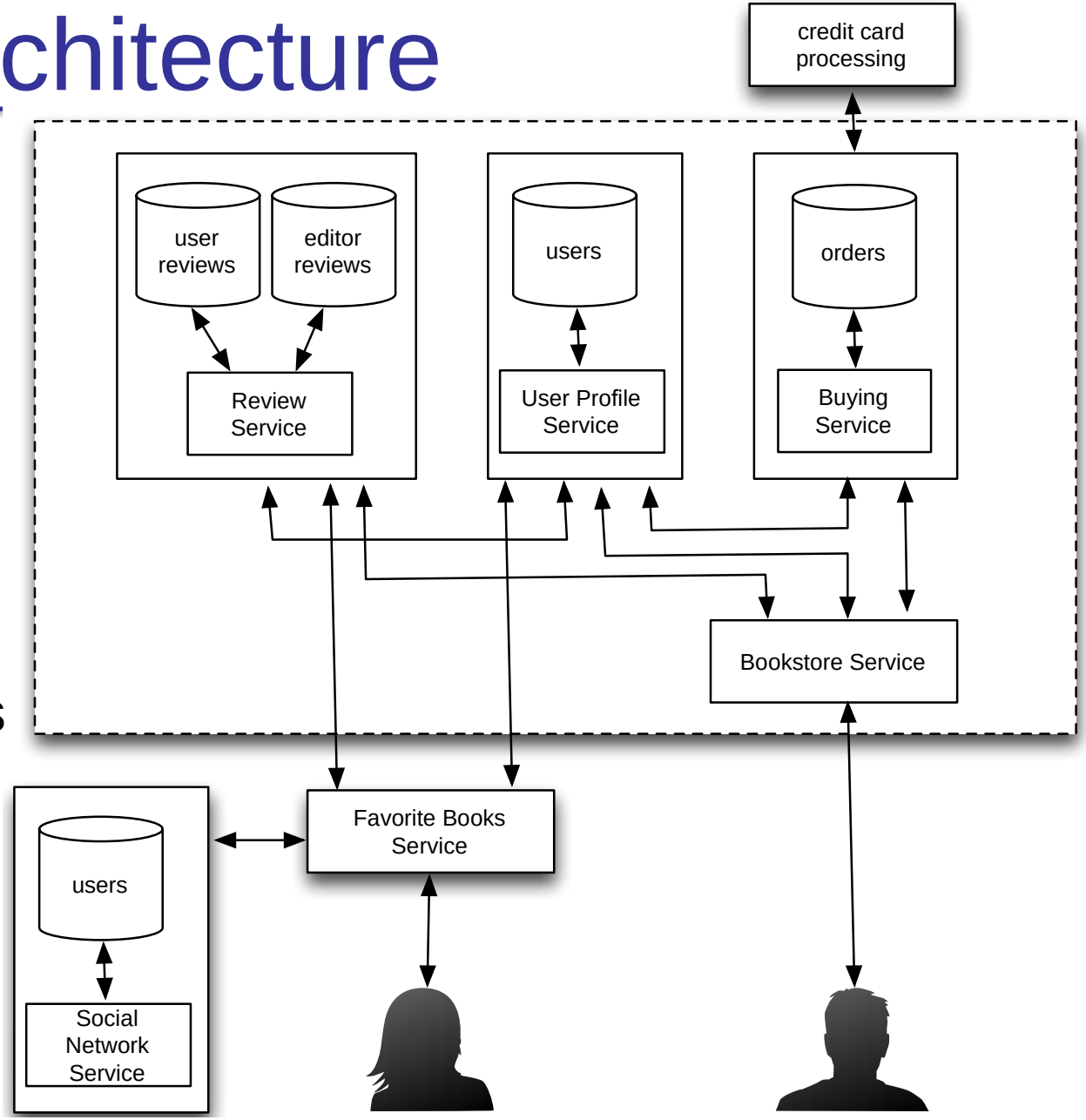
(Figure 1.2, *Engineering Software as a Service* by Armando Fox and David Patterson, 2nd Beta edition, 2013.)

Amazon CEO: thou shalt use SOA (2002, 7 years after founding)

1. “All teams will henceforth expose their data and functionality through service interfaces.”
2. “Teams must communicate with each other through these interfaces.”
3. “There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever.”
4. “Service interfaces must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. ”
5. “Anyone who doesn't do this will be fired.”

Service-Oriented Architecture

- Document-centric REST APIs
- Shared-nothing subsystems built *and maintained* by 2-pizza teams ("You build it, you run it")
  - Can recombine into new services
- Amazon later productized its own services into AWS!



(Figure 1.3, *Engineering Software as a Service* by Armando Fox and David Patterson, 2nd Beta edition, 2013.)

Picking a Platform/Target

- Platform to motivate students: smartphone or SaaS?

- Which fits better with Agile?
  - "Continuous deployment" of SaaS is natural fit

- Which has best tools to learn by doing?
  - Save time given only 4-5 weeks
  - Easier for student to follow advice through "learning by doing"
  - Can grade process vs. just final project

- Life lesson: get good at learning new tools/languages

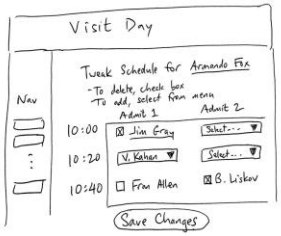
Revamped Berkeley Course

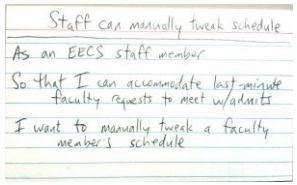
- Do SaaS project in "2-pizza teams"
- Use iteration-based Agile/XP
- Use free, world-class tools of Ruby/Rails ecosystem to support Agile processes
- Recruit non-technical customers from nonprofits
  - Grateful for help, involved stakeholders
  - Non-technical➔students must learn communication
- Focus on service-oriented architecture

Outline

- State of Software Engineering Education
- Reinventing SW Engineering with Agile+SaaS
- Description & Evaluation of Revamped Course
- Curricular Tech Transfer: MOOCs & SPOCs
- Local Lessons

2-week Agile/XP Iteration





Cucumber

PIVOTAL TRACKER

heroku

Talk to customer

Lo-fi UI mockup

User stories & scenarios

Legacy Code

Design patterns

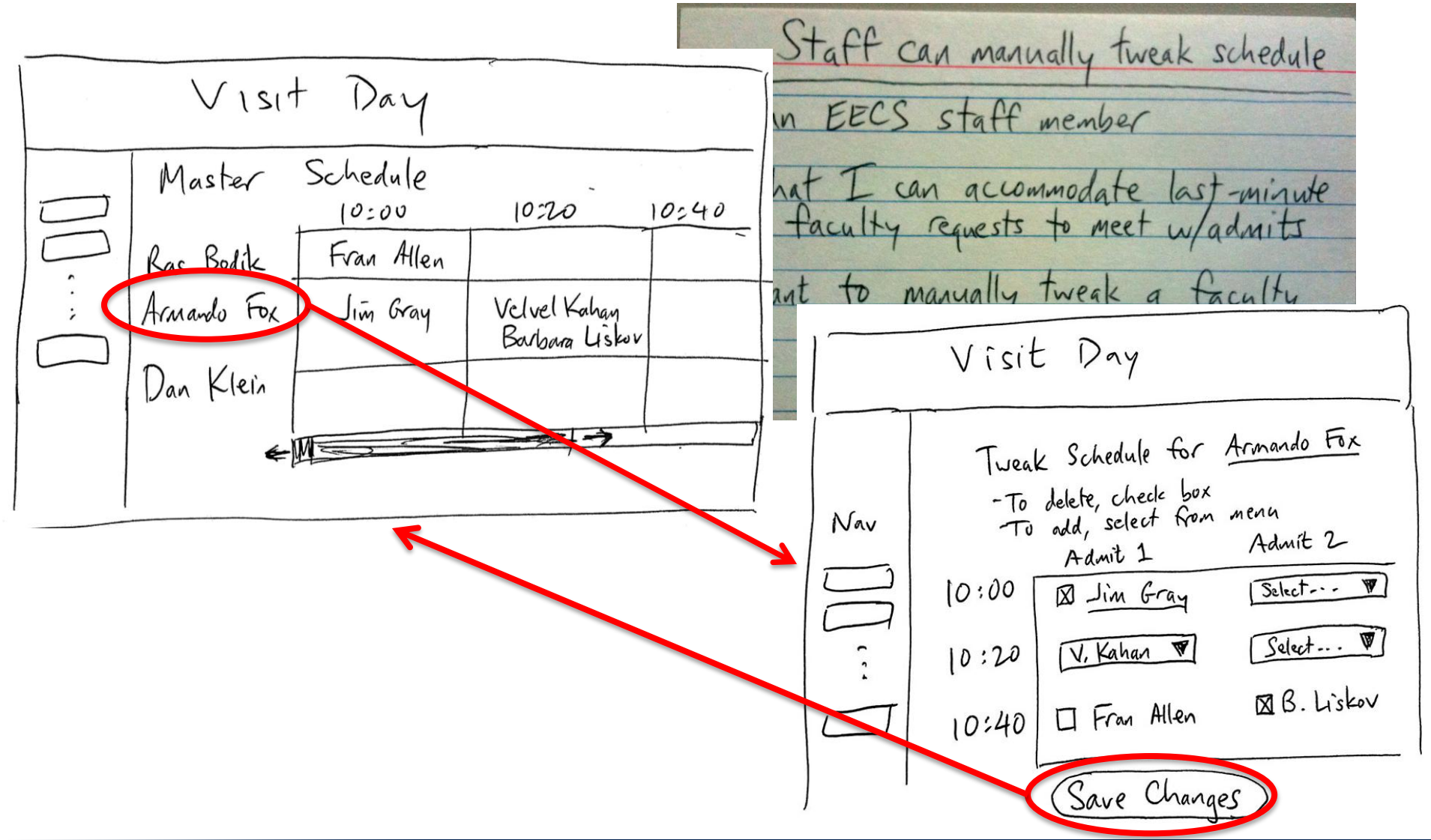
Behavior-driven Design / user stories

Test-first dev. (unit/funct.)

Measure Velocity

Deploy

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

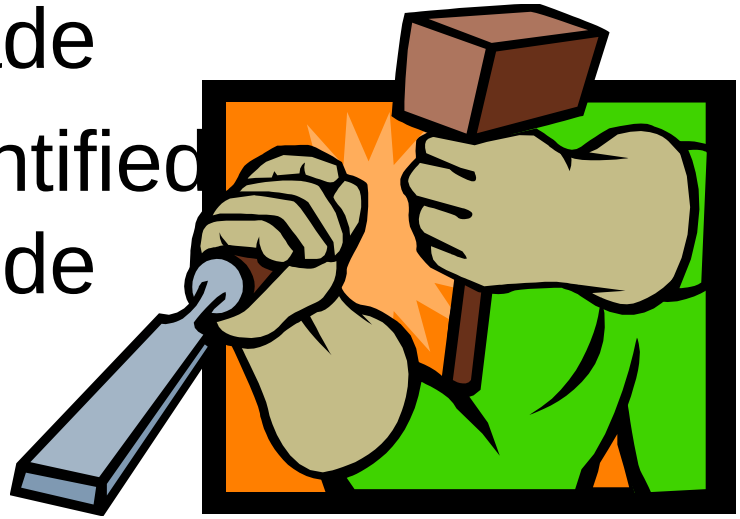
Scenario: add an admit to a meeting with an open slot
Given "Velvel Kahan" is available at 10:20
When I select "Velvel Kahan" from the menu for the 10:20 meeting with "Armando Fox"
And I press "Save Changes"
Then I should be on the master meetings page
And I should see "Velvel Kahan added to 10:20AM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

Scenario: remove admit from meeting

...

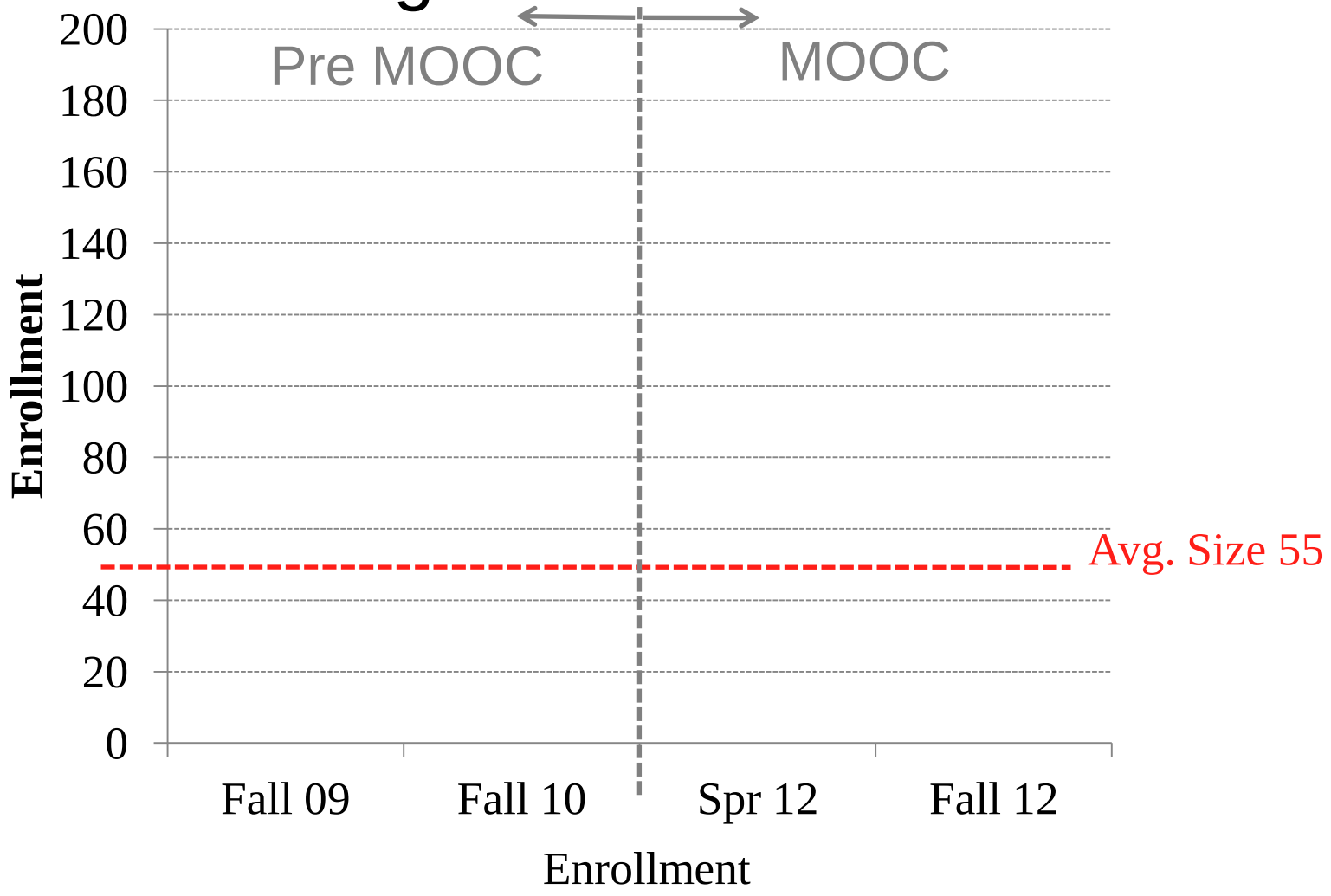
Methodologies ➔ Tools

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



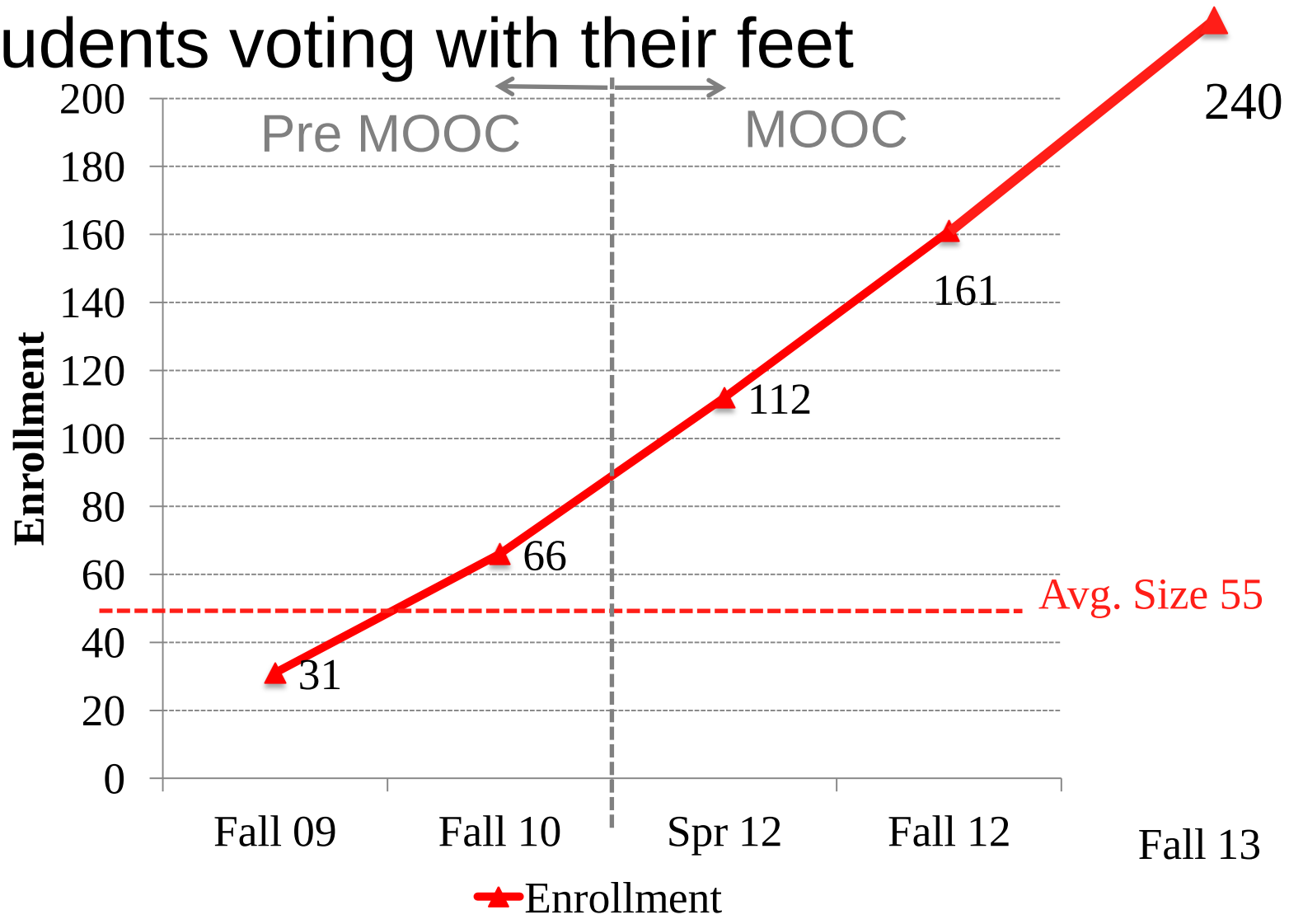
Evaluating Agile Approach

- Students voting with their feet



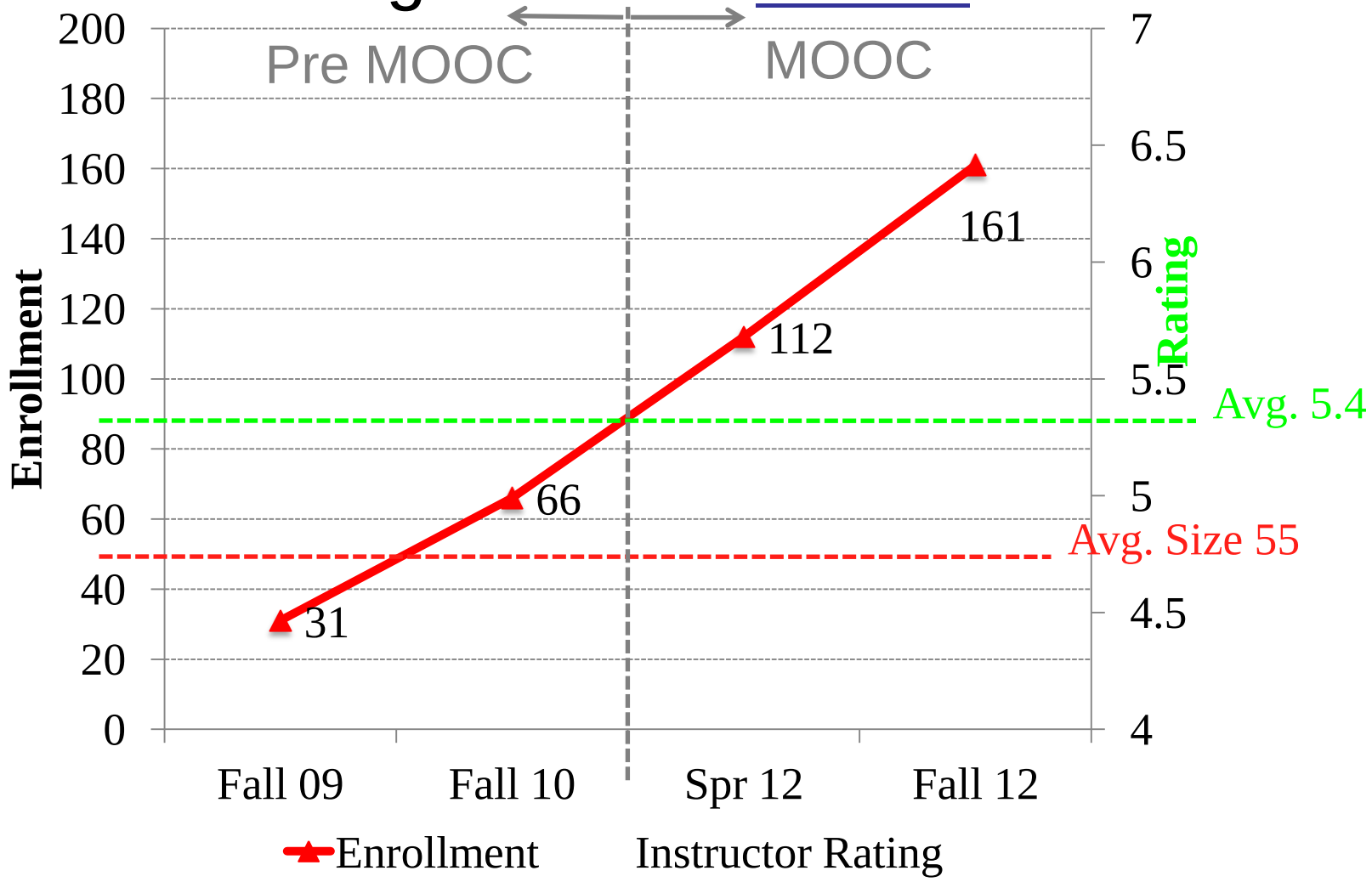
Evaluating Agile Approach

- Students voting with their feet



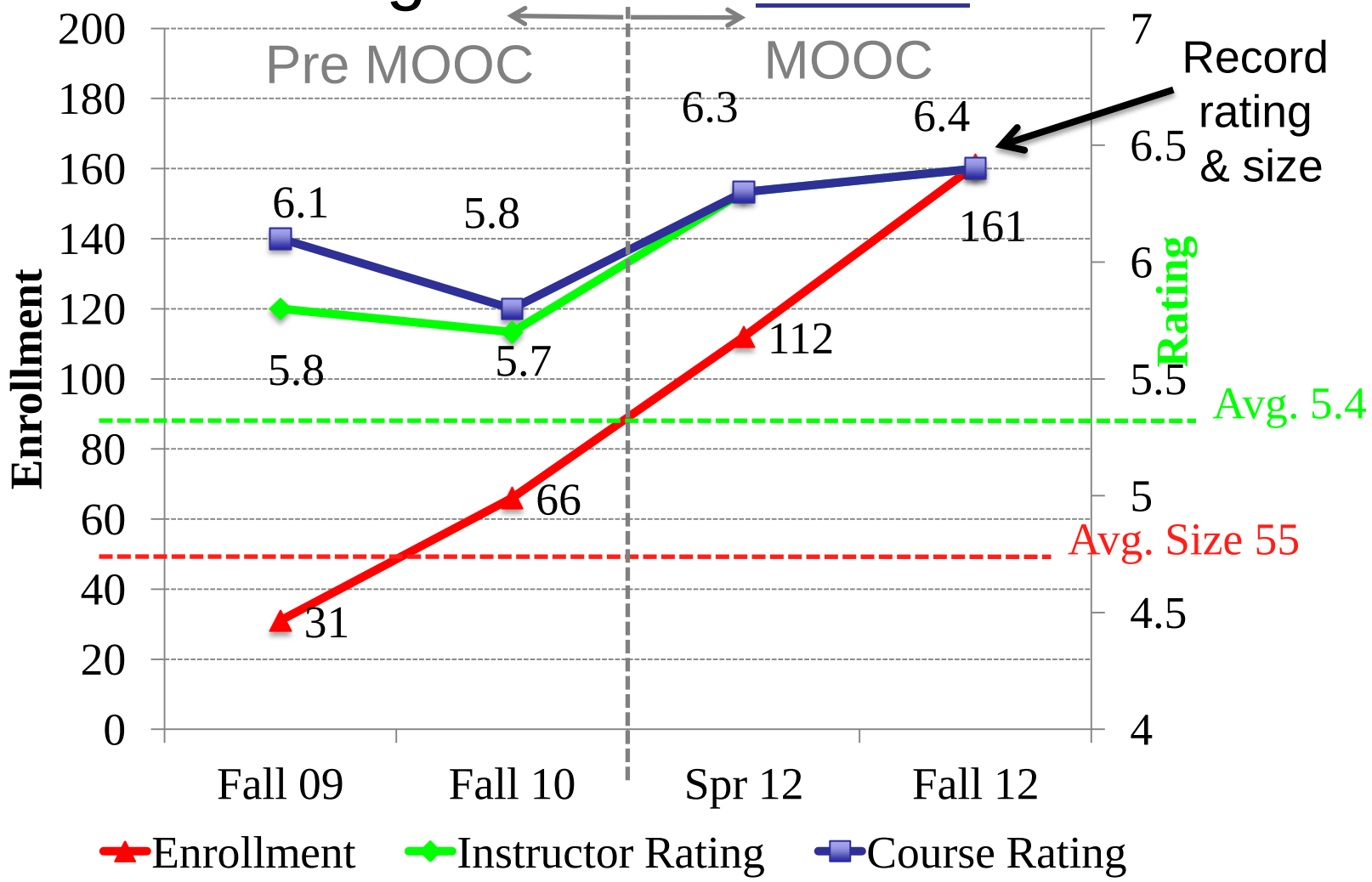
Evaluating Agile Approach

- Students voting with their **hands**



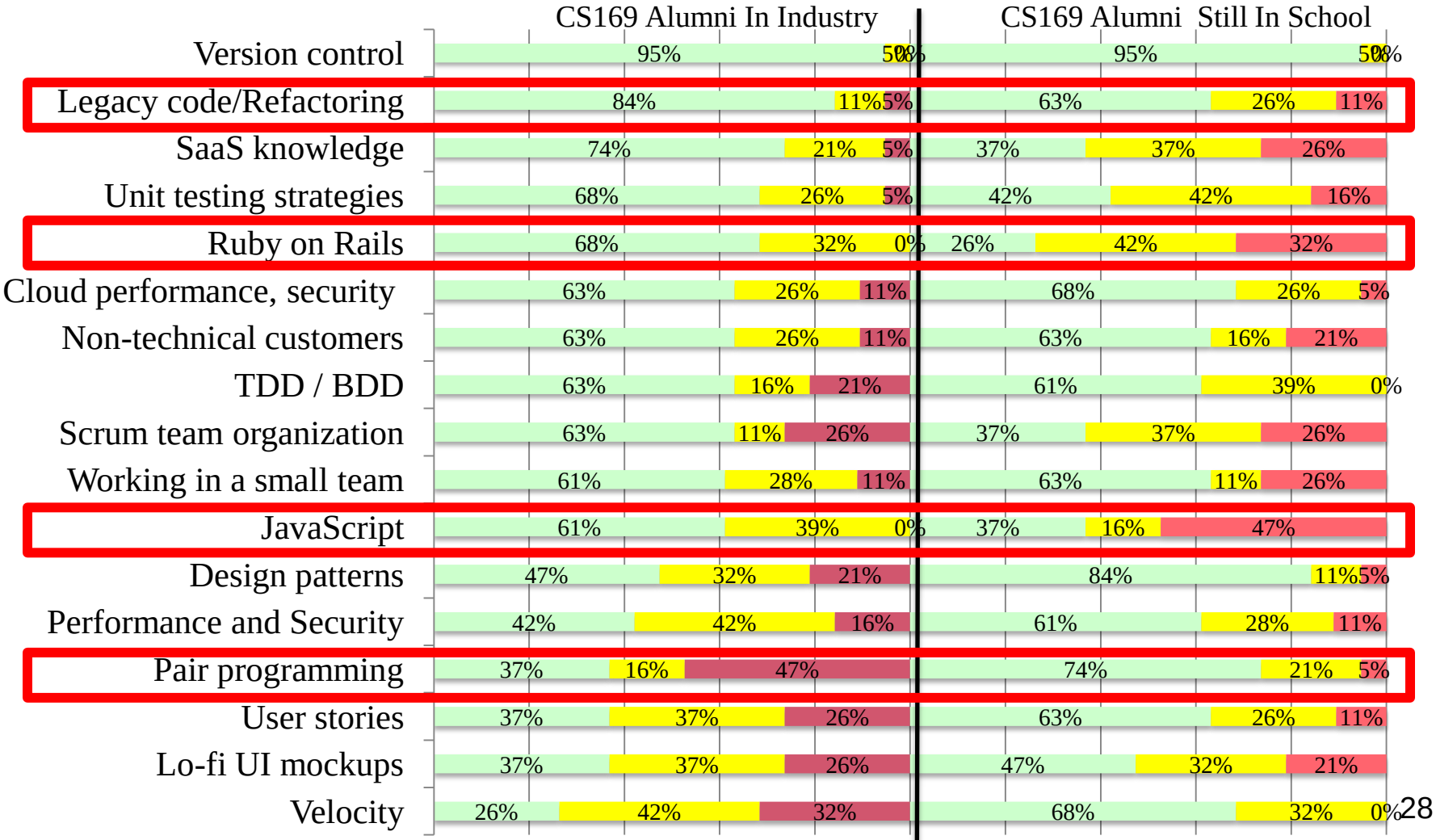
Evaluating Agile Approach

- Students voting with their **hands**



Alumni in Industry Agree

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



Employers Agree

- (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.
—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)

Customers Agree

- http://vimeo.com/44837891
- https://vimeo.com/channels/saasprojects

ACM/IEEE 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"

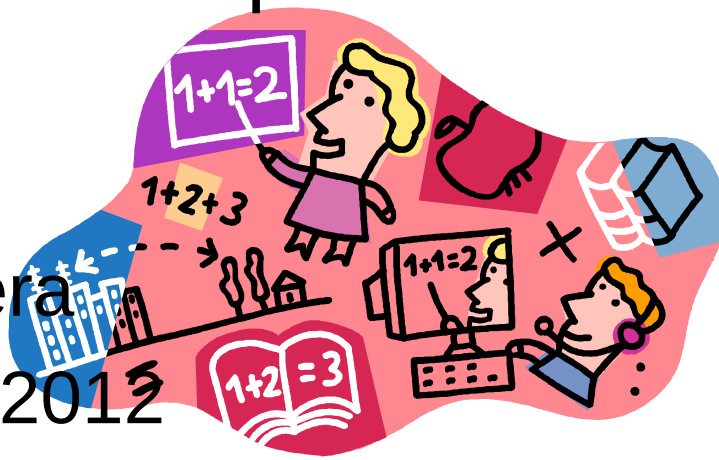
Joint Task Force on Computing Curricula, "Computer Science Curricula 2013, Ironman Draft (version 1.0)," ACM/IEEE CS, Feb. 2013

Outline

- State of Software Engineering Education
- Reinventing SW Engineering with Agile+SaaS
- Description & Evaluation of Revamped Course
- Curricular Tech Transfer: MOOCs & SPOCs
- Local Lessons

MOOC Surprise

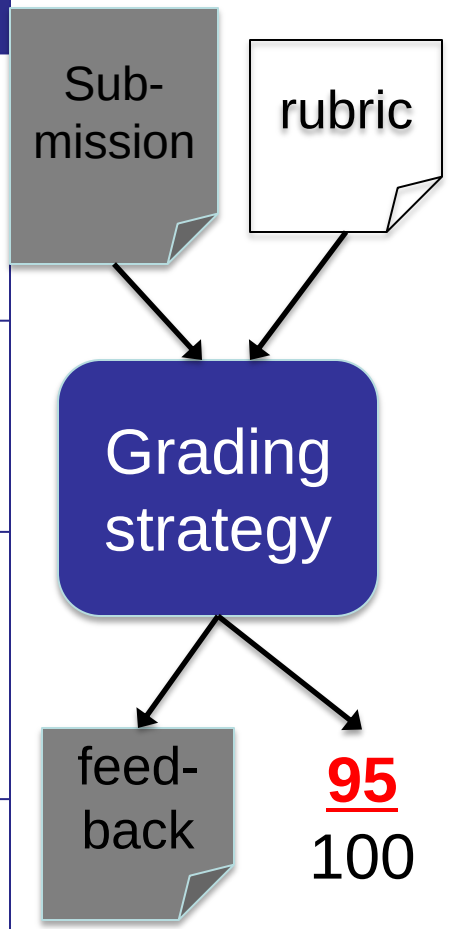
- Nov 2011: recruited for Massive Open Online Course
  - 1st Berkeley MOOC
  - 1st or 2nd MOOC from Coursera
  - 10,000 earned certificates in 2012



- Recorded MOOC 3 times since
  - Each re-record: MOOC lags Berkeley course by 4-5 weeks
  - After re-record: re-run MOOC 2-4 more times
  - Volunteer “World TAs” keep it going

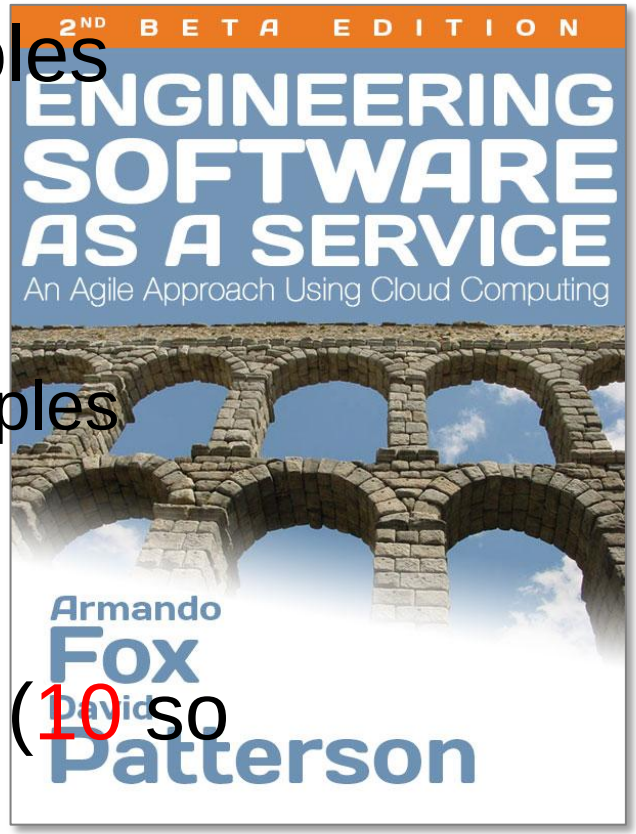
Autograding Strategies

Assignment type	Grading strategy
Write code	• RSpec (correctness) •[soon] 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] secure online quiz-taking?



Old Curricular Tech Transfer: Textbook

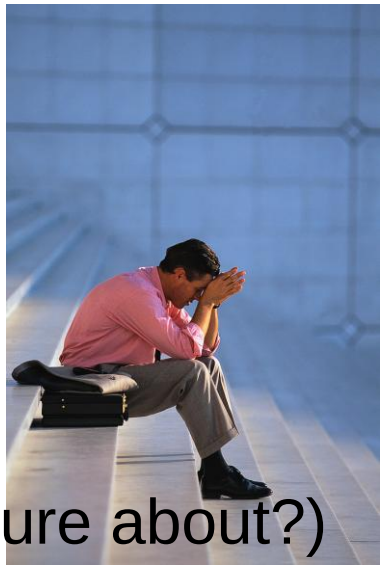
- Connects topics by their Agile roles
- Developed together with MOOC
  - 1 book chapter ⇔ 1 week of class
  - 1 section ⇔ 1 MOOC module
  - Ebook has live links to code examples & Wikipedia definitions
  - MOOC revised ⇔ ebook edition (Alpha, Beta, 2nd Beta)
- Ebook gets instant free updates (10 so far)
- Rated 4.5★/5 on Amazon



print $30
ebook $10
both $33

Profs become TAs for MOOC-based 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



5 SPOCs (Spring 2013)

Software Engineering Curriculum Technology Transfer: Lessons Learned from E-books, MOOCs, and SPOCs. A. Fox & D. Patterson, Proc. SPLASH-E 2013. http://www.saasbook.info/about

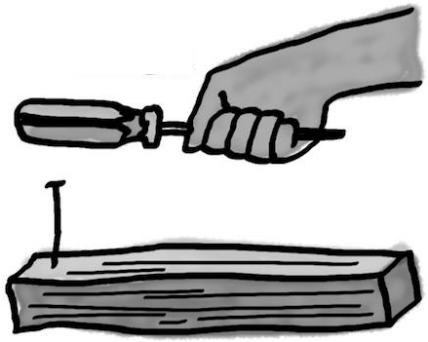
	Bingham-ton	UC Berkeley	U. Colo. Colorado Springs	Tsinghua	UNC Charlotte	Hawaii Pacific
Engineering SaaS Textbook	✔	✔	✔	✔	✔	✔
Instructor Reviews Video	✔	✔	✔	✔	✔	✔
Reuse Assignments	✔	✔	✔	✔	✔	✔
Autograde Assignments		✔	✔	✔	✔	✔
Reuse Exams	✔		✔	✔	✔	
Have Local SPOC Forum		✔		✔	✔	✔
Reuse Lecture Slides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

Are MOOCs Dead?

"I think we've found the magic formula."
Information Week, August 2013

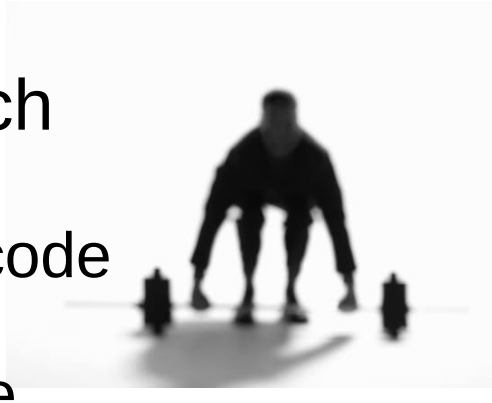
Sebastian Thrun, Founder, Udacity

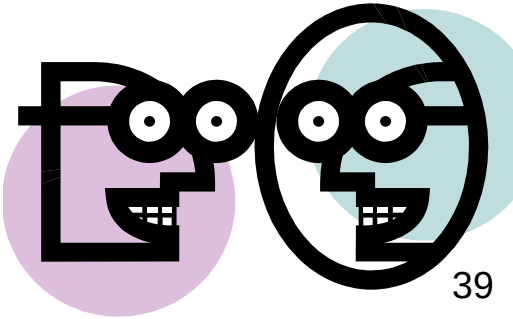
- Alternative view: misapplication of a promising technology
- SPOC + Ebook lets MOOCs perform curricular technology transfer



(Their) Conclusions

- SaaS & Agile revolutionize SW industry + make it *easier & more effective* to teach SWE in classroom
  - SaaS tools ➔multiple iterations ➔better code ➔ learn, use & appreciate SWE concepts
  - Faculty, students, & industry now embrace revised SW Eng course
- SPOCs+Ebooks expand *number of classrooms,* empower faculty, improve productivity
  - SPOC+Ebook increase *size, scope* of course & allow easier *repetitions*
  - "personalizing" MOOCs for campus





We are looking for more SPOC‘ers

Visit http://www.saasbook.info/instructors to:

- Request instructor copies
- Join instructor discussion group
- Read Instructor Guide that goes with book
- Read about our experience (op-eds, experience reports)
- Read about ACM/IEEE CS 2013 Curriculum Guidelines
- Get VMimage prepopulated for coursework
- Sign up for a SPOC…it’s free (for now anyway)

Acknowledgments: Staff of UC Berkeley CS 169, support staff for EdX CS 169.1x/169.2x, World TAs, SPOC instructors (esp. Sam Joseph)

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* 56:12, Dec. 2013
- 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.

Tutorial for Instructors

- For instructors interested in adopting ESaaS in their classrooms
- Explanations and demos:
  - Overview of syllabus, tools, correspondence with MOOC
  - Demos: Assignments, video lectures, online quizzes
  - Demos: Autograders for programming assignments
  - edX SPOC interface for instructors

Outline

- State of Software Engineering Education
- Reinventing SW Engineering with Agile+SaaS
- Description & Evaluation of Revamped Course
- Curricular Tech Transfer: MOOCs & SPOCs
- Local Lessons

SPOC-IL

- 3 Hour meeting per week elective course
  - Time manag.: materials / discussion / quiz / hw mentoring / project meetings
  - 4th year SE students, alumni of SE course (and working on final project)
- Textbook & videos hardly used (language?)
- Peer Evaluations

Some Lessons

- “Open source” solutions
- Balance Project / Homework
- Project progress vs. following best practices, e.g. TDD, fixed iterations
- Tools: VM, Git&github, IM, Heroku, PM
- Edx Platform issues

Projects

Group	repo.	pivotal	prod.
1-psifas	https://github.com/liorzaken/Psifas	https://www.pivotaltracker.com/projects/1224912	http://psifas.herokuapp.com/
2-animal-assist	https://github.com/Benoh/SAAS-Project-Animal-Assist	https://www.pivotaltracker.com/projects/1224920	https://animal-assist.herokuapp.com/
3-etnachta	https://bitbucket.org/EladYona/etnachta	https://www.pivotaltracker.com/projects/1224926	https://etnachta.herokuapp.com/
4-speech-therapy	https://github.com/ISGO/SpeechTherapy	https://www.pivotaltracker.com/projects/1224928	http://speech-therapy.herokuapp.com/

Thanks

BACKUP SLIDES

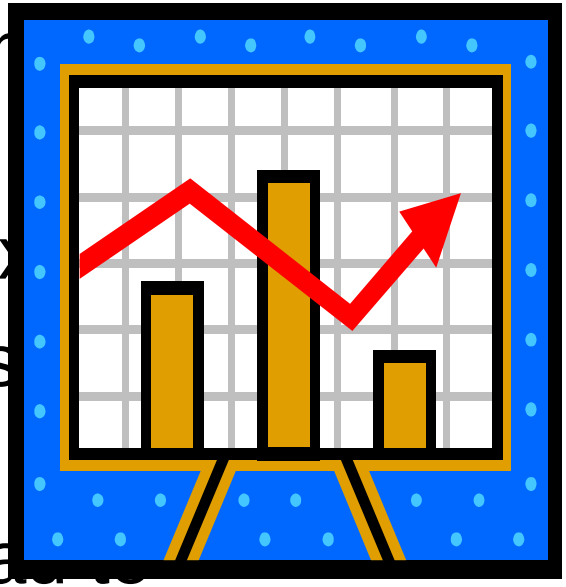
Replace courses with MOOCs?

- "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

MOOCs weaken on-campus pedagogy?

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



- Reality: can help to **improve** on-campus pedagogy

MOOCs distract faculty & hurt productivity?

- Berkeley: 4X students in SW Eng course
- SJSU tried EE MOOC from M
  - MOOC homeworks, lectures
  - Same exams as prior SJSU co
  - 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

SPOC Good & Bad

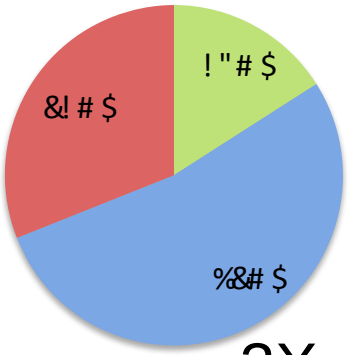
- Auto-graders took grading burden off staff & emphasized TDD
- Video lectures dense, efficient (can rewind)
- Students excited about latest tech (Rails, Agile)
- Students impressed with “world-class” instruction

- Given MOOC, answers available on Internet
- Auto graders did not check code style (fixing now)
- Some student computers too slow to run VM
- Some didn’t know Linux
- Some got

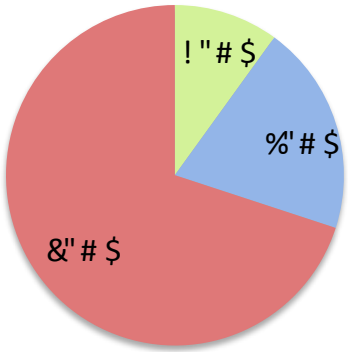
SPOCs Next Time

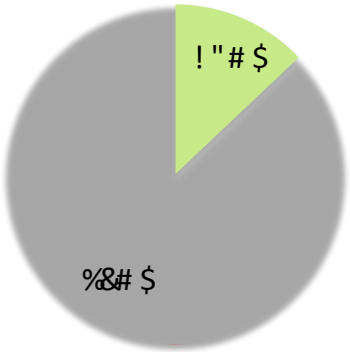
- Try participating in global MOOC Forum?
  - Talk to students other schools about common problems
  - Broaden SWE perspective
  - Access to “World TAs” to help with technology (Ruby, Rails, tools)
- All 5 repeating this semester

How Well do Plan-and-Document Processes Work?



3X





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

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%),

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%

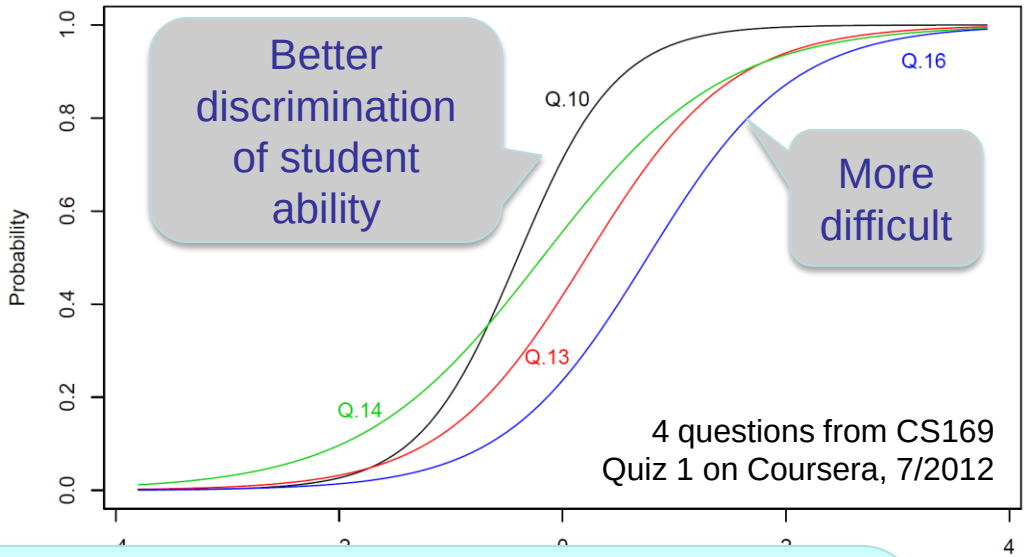
- Reality: Threatens **continuing education** programs

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 co
- 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)

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*

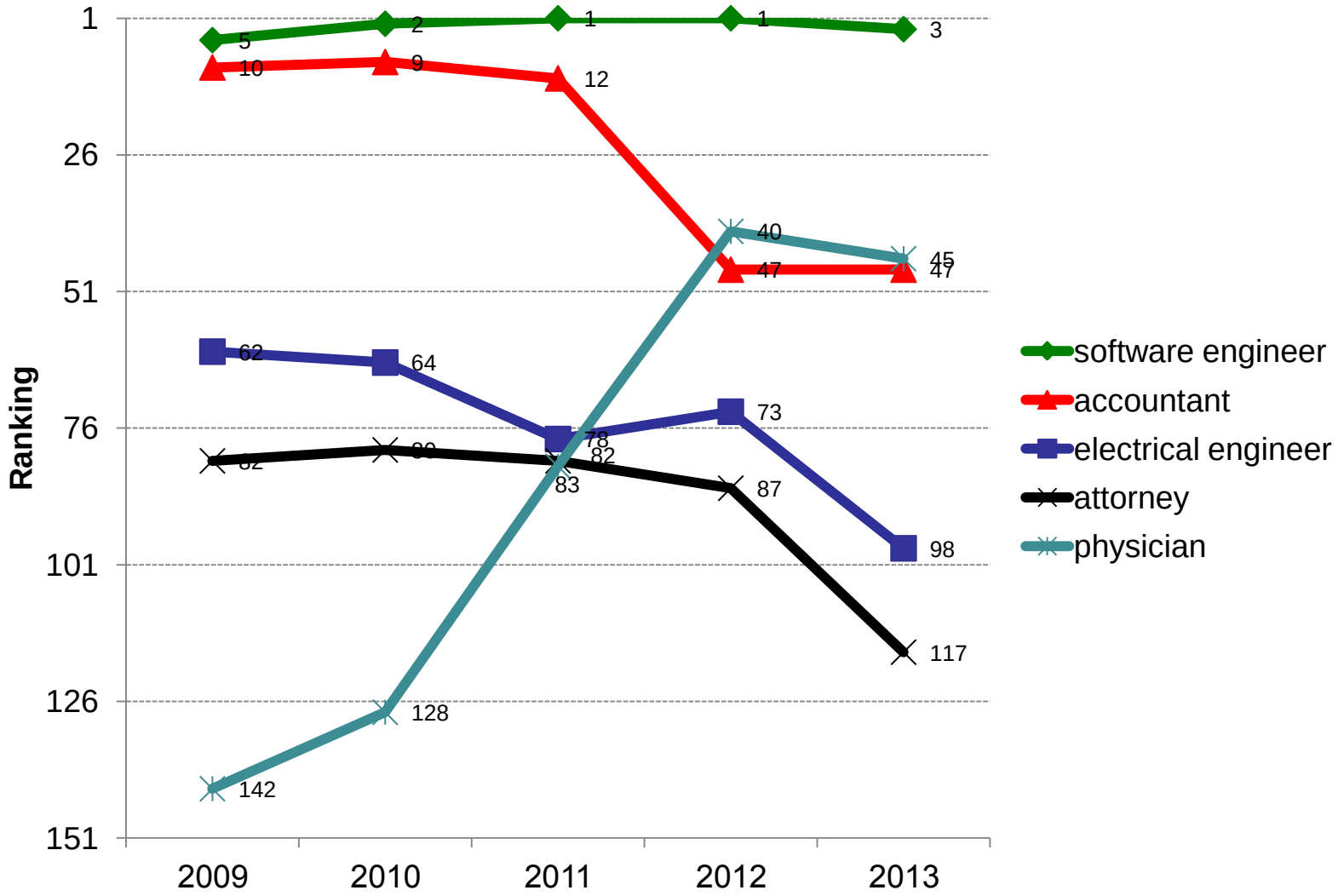
Joint Task Force on Computing Curricula, "Computer Science Curricula 2013, Ironman Draft (version 1.0)," ACM/IEEE CS, Feb. 2013

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*

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.)

Methodologies ...become Tools

- Software arch., design patterns, coding practices
- Test-first development, unit testing
- Behavior-driven design, integration testing
- Agile, iteration-based project management
- Version management & collaboration skills
- SaaS technologies, deployment & operations

- Ruby & Rails
- RSpec
- Cucumber
- Pivotal Tracker
- Git & Github
- Cloud computing: EC2, Heroku

Methodologies Become Tools

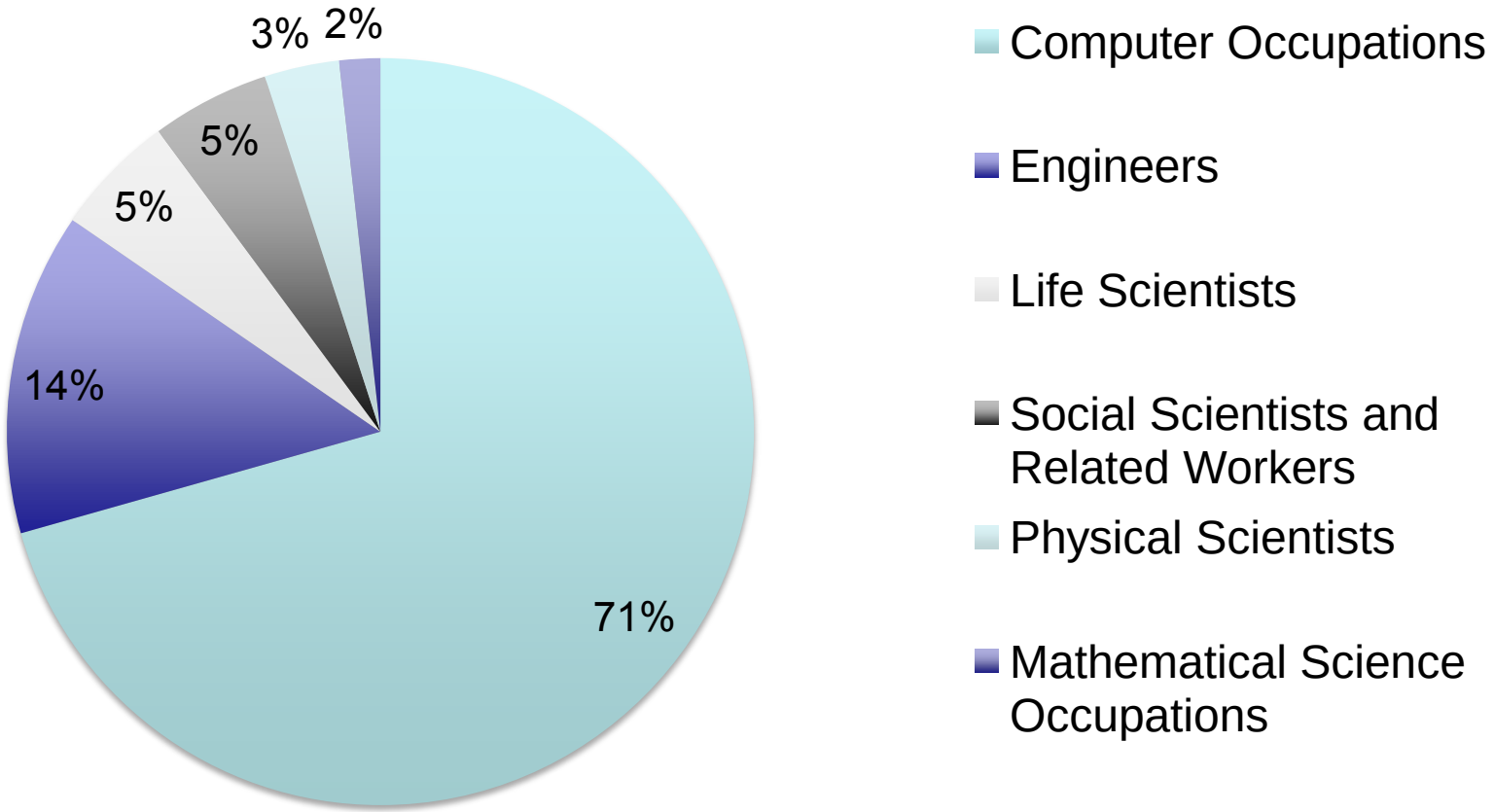
- Runs “natural language” user stories as integration tests
- Each *scenario* describes one user story
  - *Given* steps*:* setup preconditions
  - *When* steps*:* take actions, using built-in browser simulator
  - *Then* steps*:* assertions to check post-conditions
- *Step definitions* match story steps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

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
- Bay Area Mountain Rescue
- CalTeach @ Berkeley
- Bonus: New CS student outreach organization

Myth: No US Software Jobs

STEM Job Growth 2010 to 2020, US Bureau of Labor Statistics



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. 6 minute segments >> 60 minute lectures
   - Khan Academy "discovered" 10 minute segments
   - Recent MOOC research: 6 minutes median viewing time

2008 4. Scaled up question answer: By students & TAs in online forums (like StackOverflow.com)

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