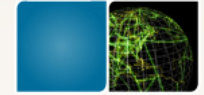


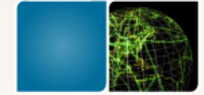
# CS Research on MOOCs and Online Education

- Amy Bruckman (organizer)
  - Professor of Interactive Computing, Georgia Tech
- Marti Hearst
  - Professor of Computer Science, School of Information, UC Berkeley
- Rob Miller
  - Professor of Computer Science and Engineering, Massachusetts Institute of Technology
- Scott Klemmer
  - Associate Professor, Cognitive Science and Computer Science & Engineering

# Computer Science Research on Higher Education?



- Future of online education is not just about trying things out
  - Research needed
  - Interdisciplinary
    - Industry
    - Schools of education
    - Schools of computer science
- How do we support this new interdisciplinary practice?



# Intriguing Research Challenges

- Three strong examples of CS research on online education
- Discussion of the challenges
- Discuss both MOOCs and distance ed
  - Make clear which issues refer to which
- Thanks to Andy, Greg, and Brent

# Designing a World that Teaches Itself



**Scott Klemmer**

Cognitive Science + Computer Science & Engineering

UC San Diego + Stanford



CHALLENGE & OPPORTUNITY

# Design at Large









# RESEARCH EXAMPLES



peer assessment



small-group discussions



richer feedback



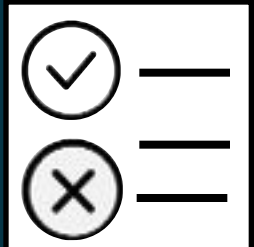
Predict



Identify



Verify

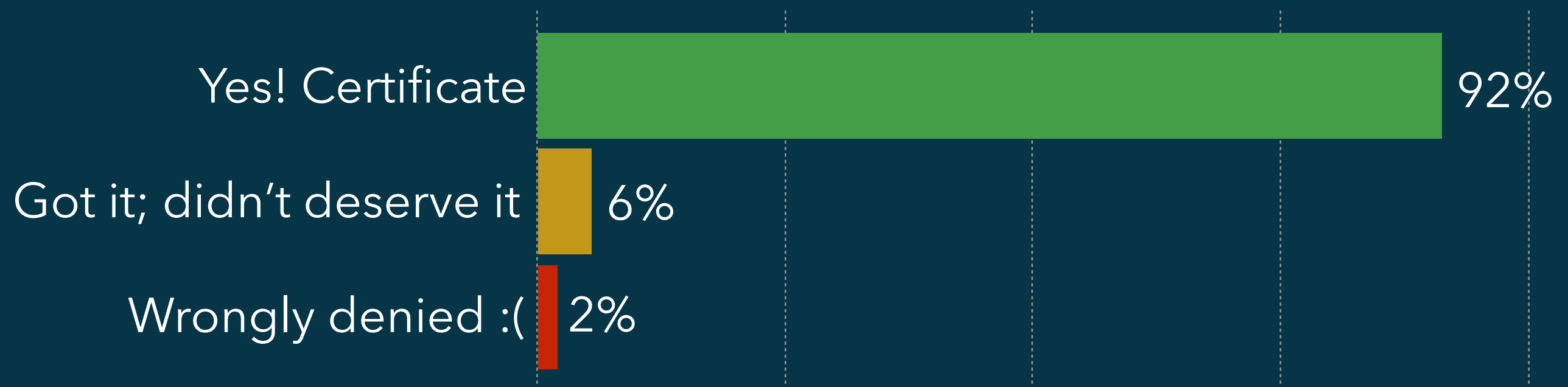


Results

machine+peer learning

AT LARGE...

# Peer assessment



AT LARGE...

# Peer assessment in 100+ classes



Human-computer  
Interaction  
*Design*



Programming  
in Python  
*Code*



Introduction to  
Philosophy  
*Essays*



Teaching  
character  
*Management*



Child  
Nutrition  
*Recipes*



Social  
Psychology  
*Essays*



Constitutional law  
*Arguments*



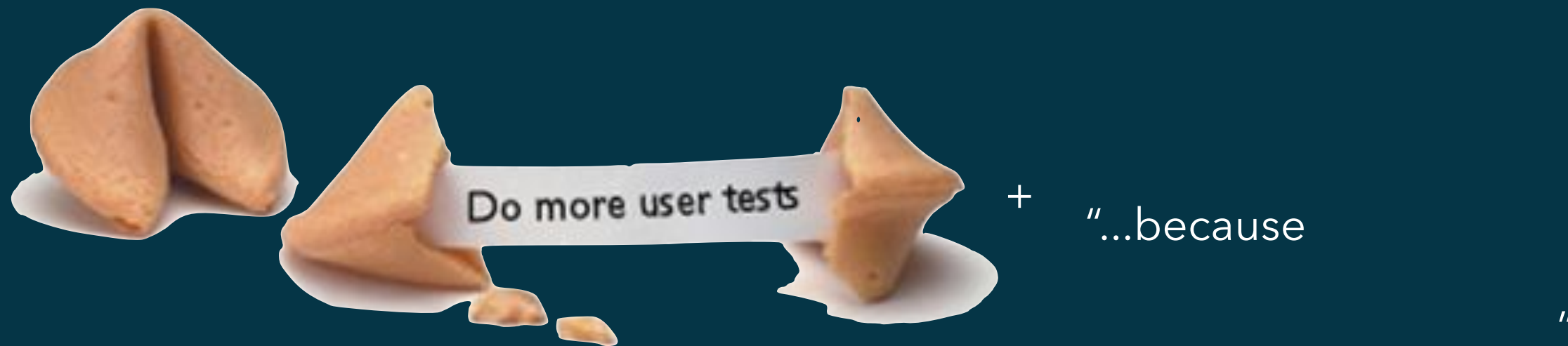
World Music  
*Music*



## FORTUNE COOKIES

# Qualitative, personalized feedback

- Peers can recognize errors from a list of patterns, even if they can't articulate them
- Most errors are variations on a theme





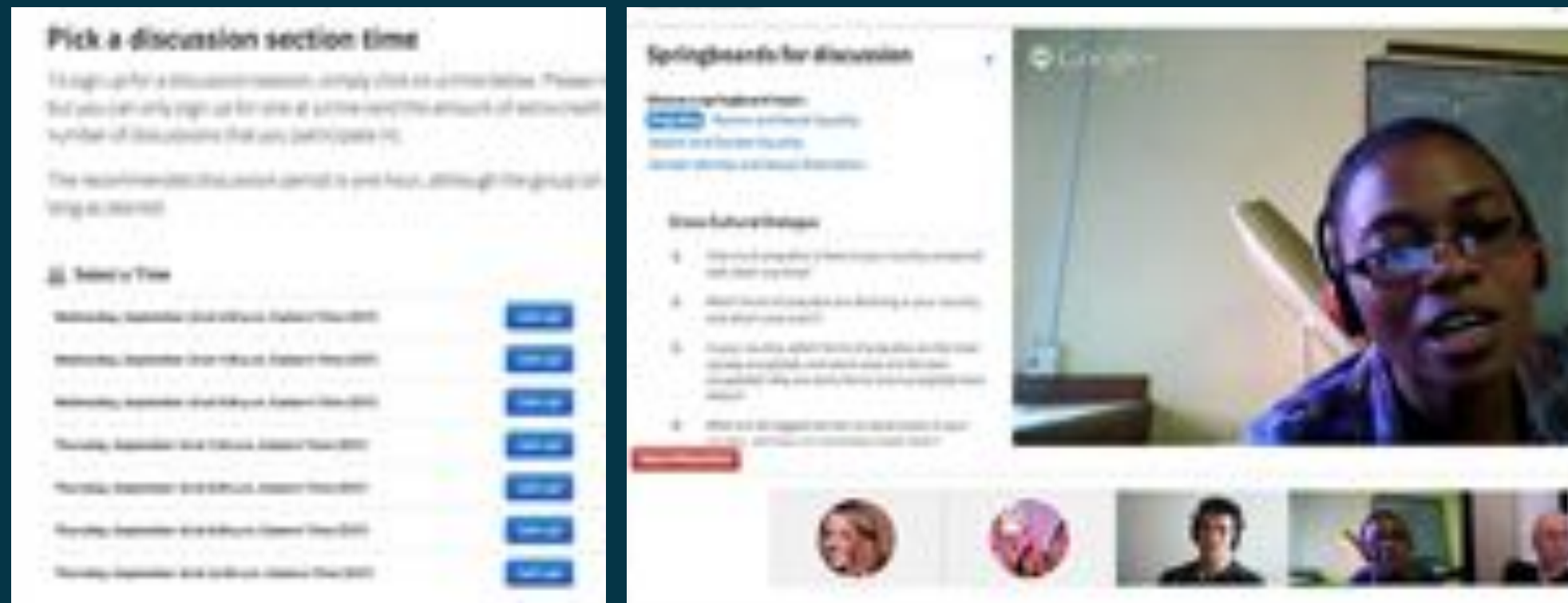
# Alone Together?





# LEVERAGING DIVERSE EXPERIENCES

## Small groups in massive classes



“It was like a mini-UN. We had an Australian currently residing in Dubai, an Afghan, a Romanian, an Indian & myself (a Pakistani).”



IDENTIFY-VERIFY

# Creating Micro-Experts

- richer semantics increase quality

from scores

• \_\_\_\_\_

• \_\_\_\_\_

• \_\_\_\_\_



to labels

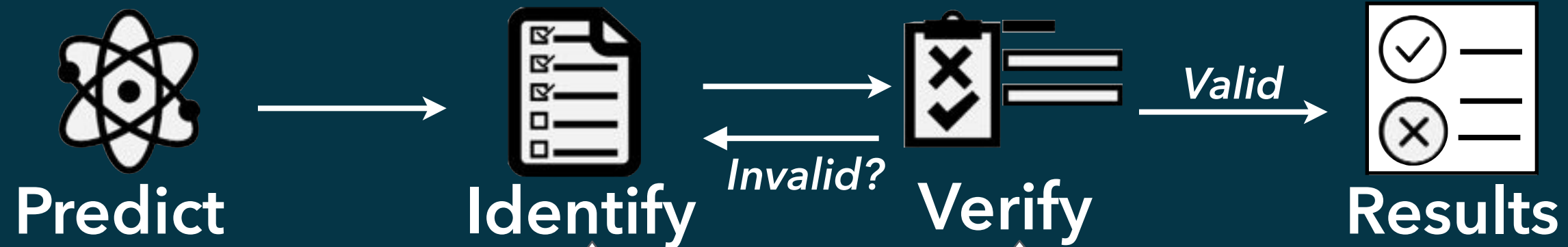
☐ \_\_\_\_\_

☒ \_\_\_\_\_

☒ \_\_\_\_\_

# IDENTIFY-VERIFY

## Machines modulate peer grading



Answer guide: In general, answers should mention benefits of sharing multiple prototypes. Answers that only mention the benefits of sharing one prototype should not receive credit.

**Student answer:** 1) More Creativity in the final design.  
2) Can take all the good features in different designs to make a better one.

Below, choose which attributes apply to this answer—you can choose both correct and incorrect attributes, which may result in partial credit.

**First, check if the answer has any incorrect attributes**

Here are some common attributes of an incorrect answer. Select ones that apply.

- ☐ Lower cost/investment in making designs. (This is incorrect because multiple designs often cost more to make, and we're interested in benefits of sharing, rather than making prototypes)
- ☐ Other incorrect/irrelevant answer

Student answer	correct?
These were marked as: More sharing of features between designs.	Assessment correct?
more feedback, multiple options, better creativity	<input type="radio"/> Yes <input type="radio"/> No
These were marked as: Creates increased group rapport/increased conversational turns. Both lead to better discussions.	Assessment correct?
Encourages group loyalty Produces more examples/prototypes It places the focus on the artifact and eliminates egos	<input type="radio"/> Yes <input type="radio"/> No
more feedback, multiple options, better creativity	<input type="radio"/> Yes <input type="radio"/> No

*Scaling Short-answer Grading by Combining Peer Assessment with Algorithmic Scoring,*  
Kulkarni, Socher, Bernstein, & Klemmer, Learning at Scale, 2014



## CS RESEARCH OPPORTUNITY

- Build practical theory with real-world experiments
- Bake pedagogy into software that transforms learning

<http://d.ucsd.edu/peer>





“Nothing is as practical  
as a good theory”

“The best way to  
understand something  
is to try and change it”

*–Kurt Lewin*



- Build practical theory with real-world experiments
- Bake that theory into software that transforms  $\langle X \rangle$



Real  
experiments  
are  
critical



# We Need to Do These 3 Things

- Insure that learners understand their role in experiments they opt in to  
*Good design is key, and nuanced*
- Insuring broad research access to conducting experiments, evaluating data, & open science  
*Chairs: you have an important role here*
- Few current CS curricula don't teach experimental design. More should.  
*Especially in data/HCI/learning tracks*

<http://cs303.stanford.edu>

# We Have Resources for You

- Open-source platforms with analytics, course materials, instructor resources, & graduating students :)

<http://d.ucsd.edu/peer>

# The Big Research Opportunity

- Tomorrow's online class won't look like today's (I hope)



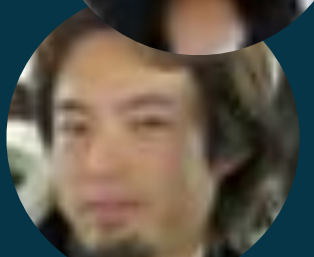
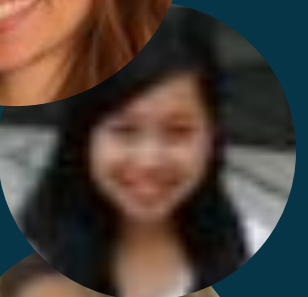
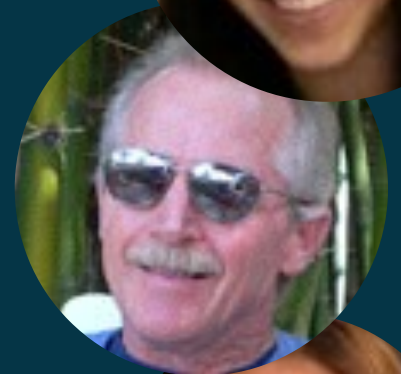
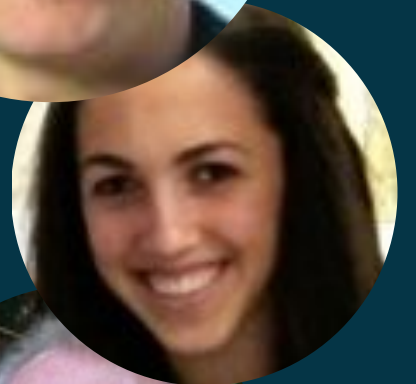
scale personalized mastery-  
learning experiences?

# Why CS?

- The scientific opportunities are tremendous
- Concrete problems are a great forge for fundamental insights
- A proud history of lifelong learning
- The CS legacy: don't just understand the world, make it a better place



with Chinmay Kulkarni  
+ many collaborators



<http://d.ucsd.edu/srk>

 @DesignAtLarge

follow student work at #hci5

# Online Education with Learnersourcing

---

Rob Miller

User Interface Design Group

MIT CSAIL

Joint work with Juho Kim, Sarah Weir,  
Elena Glassman, Philip Guo, Carrie Cai,  
Max Goldman, Phu Nguyen, Rishabh Singh, Jeremy Scott

# MOOCs: a New Scale for Learning



classroom 1:10



lecture 1:100



stadium 1:10,000

- Big problem
  - we're very far from 1-on-1 mastery learning
  - little human feedback, mass production instead of personalization, high attrition rates
- Huge opportunity
  - much faster controlled experimentation & iterative improvement
  - big online crowds can do amazing things by themselves

# Crowdsourcing vs. Learnersourcing

---

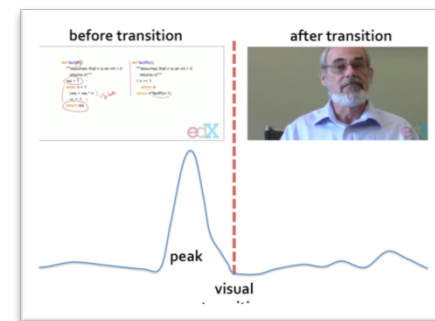
- Crowdsourcing
  - asking a crowd to do micro-work for problems we can't solve with software
  - what does the crowd get in return? money, fun, social
- Learnersourcing
  - asking **students** to do micro-work for an online course
  - what do students get in return? **learning** (hopefully)
- Types of learnersourcing
  - active: asking people to do something
  - passive: watching what people do and inferring something
- Discussion forums are active learnersourcing
  - and without them, our current MOOCs would utterly fail

# A Few Examples from My Group



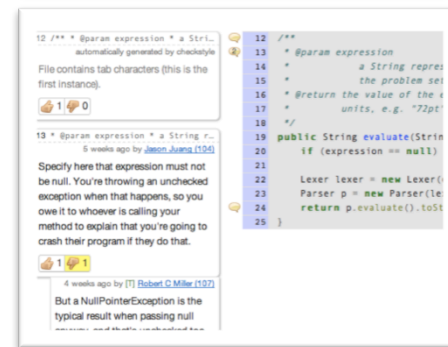
## Lecture video analytics

- find bugs and key parts in lecture videos
- passive learnersourcing



## • Peer code review

- students give feedback to each other
- active learnersourcing



## • Solution analytics

- understand the range of solutions to a coding assignment
- passive learnersourcing

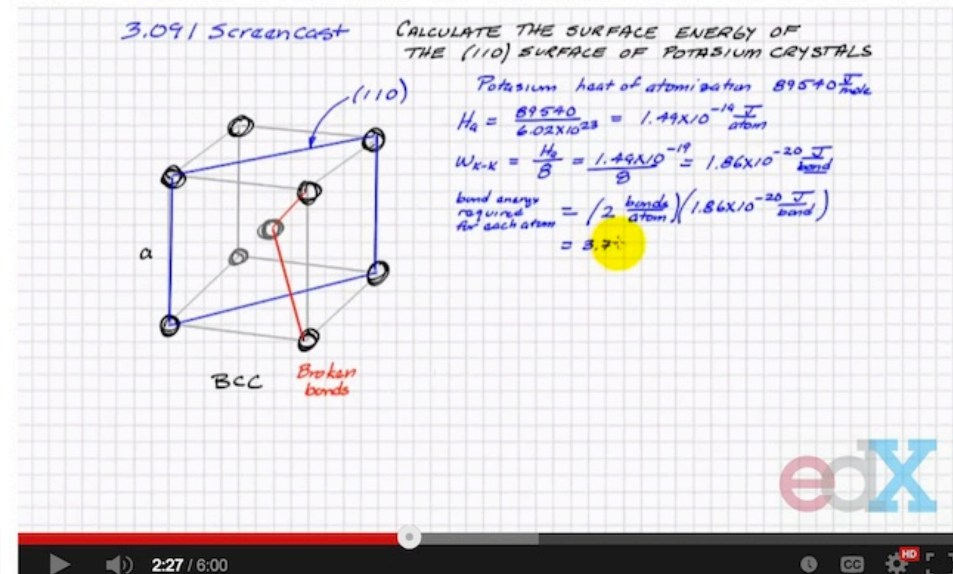
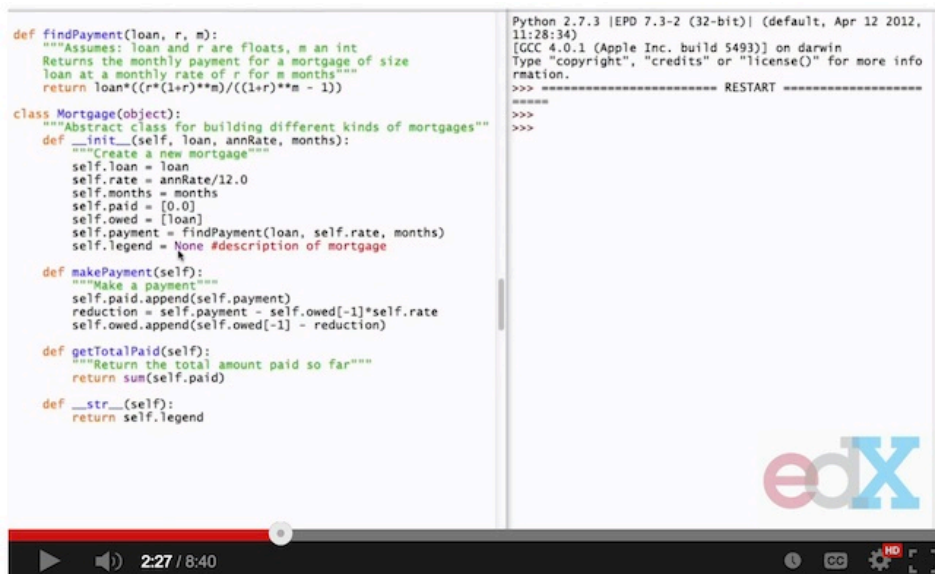
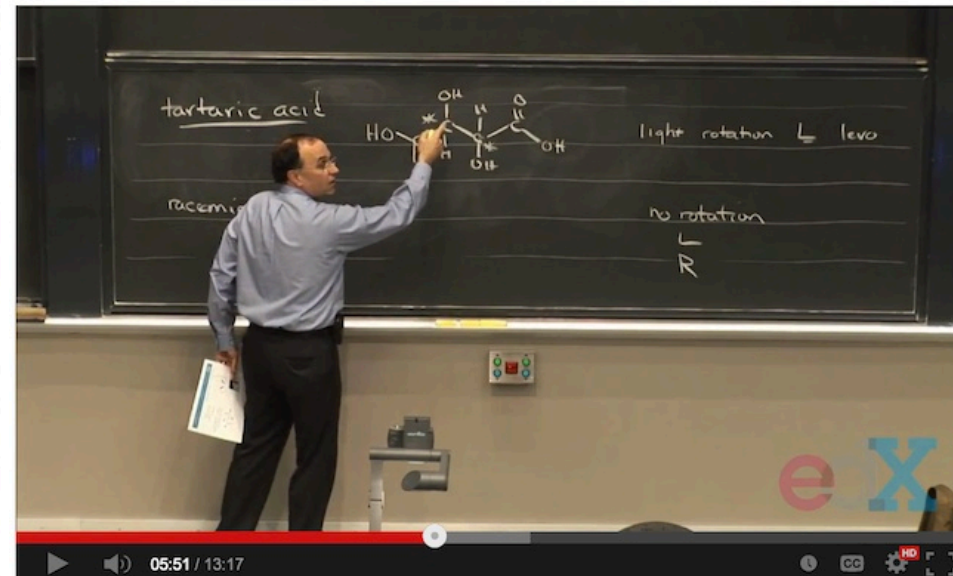
# LECTURE VIDEOS



**Juho Kim**



# MOOC lecture videos

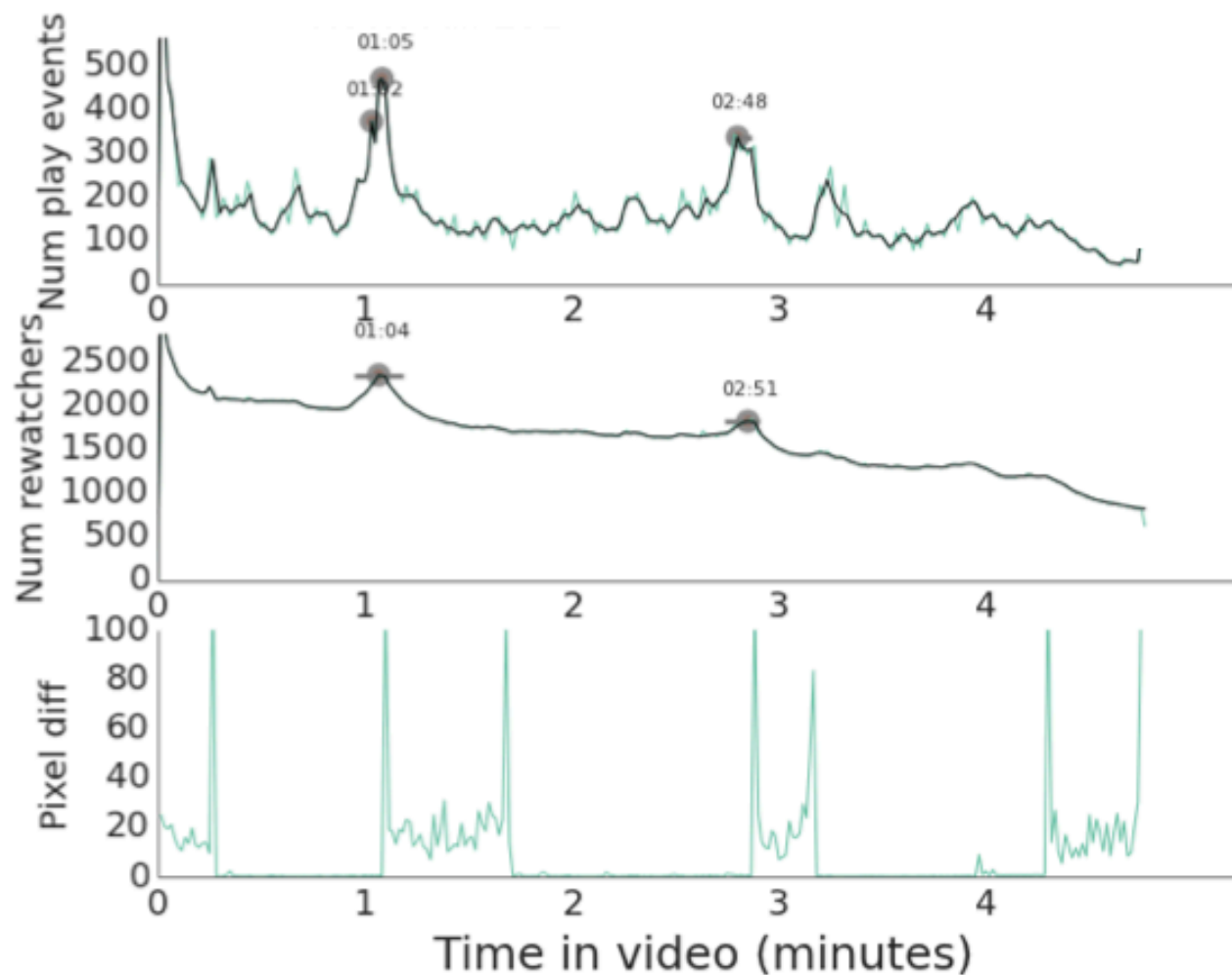


# Challenge for instructors/editors

- Don't know how students use lecture videos
  - Confusion
  - “Aha” moments
  - Bored
  - Re-watching important parts
- We analyzed video interaction data from the lectures in 4 edX courses
  - Clickstream (play, pause, scrub)

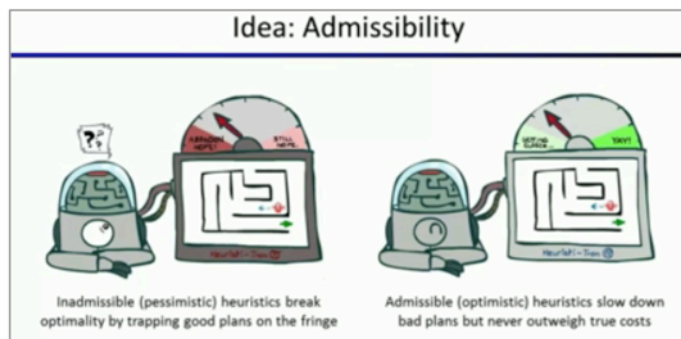
Course	Subject	University	Students	Videos	Video Length	Processed Events
6.00x	Intro. CS & Programming	MIT	59,126	141	7:40	4,491,648
PH207x	Statistics for Public Health	Harvard	30,742	301	10:48	15,832,069
CS188.1x	Artificial Intelligence	Berkeley	22,690	149	4:45	14,174,203
3.091x	Solid State Chemistry	MIT	15,281	271	6:19	4,821,837
<b>Total</b>			<b>127,839</b>	<b>862</b>	<b>7:46</b>	<b>39,319,757</b>

# Interaction Peaks



# Example: Beginning of new material

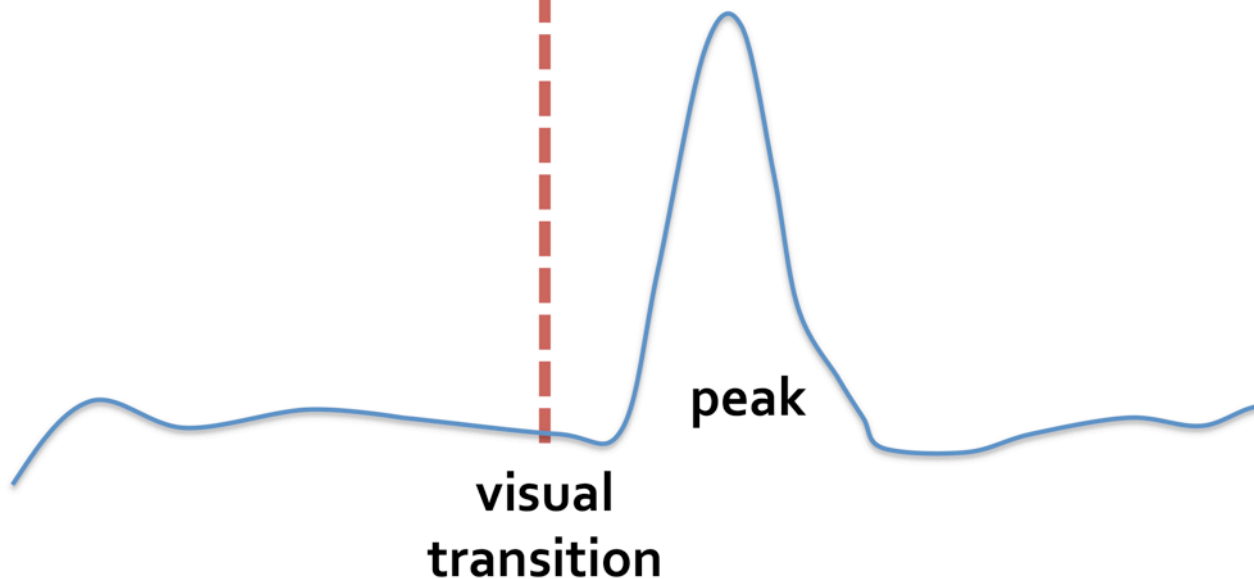
## before transition



## after transition

Admissible Heuristics

- A heuristic  $h$  is *admissible* (optimistic) if:
 
$$0 \leq h(n) \leq h^*(n)$$
 where  $h^*(n)$  is the true cost to a nearest goal



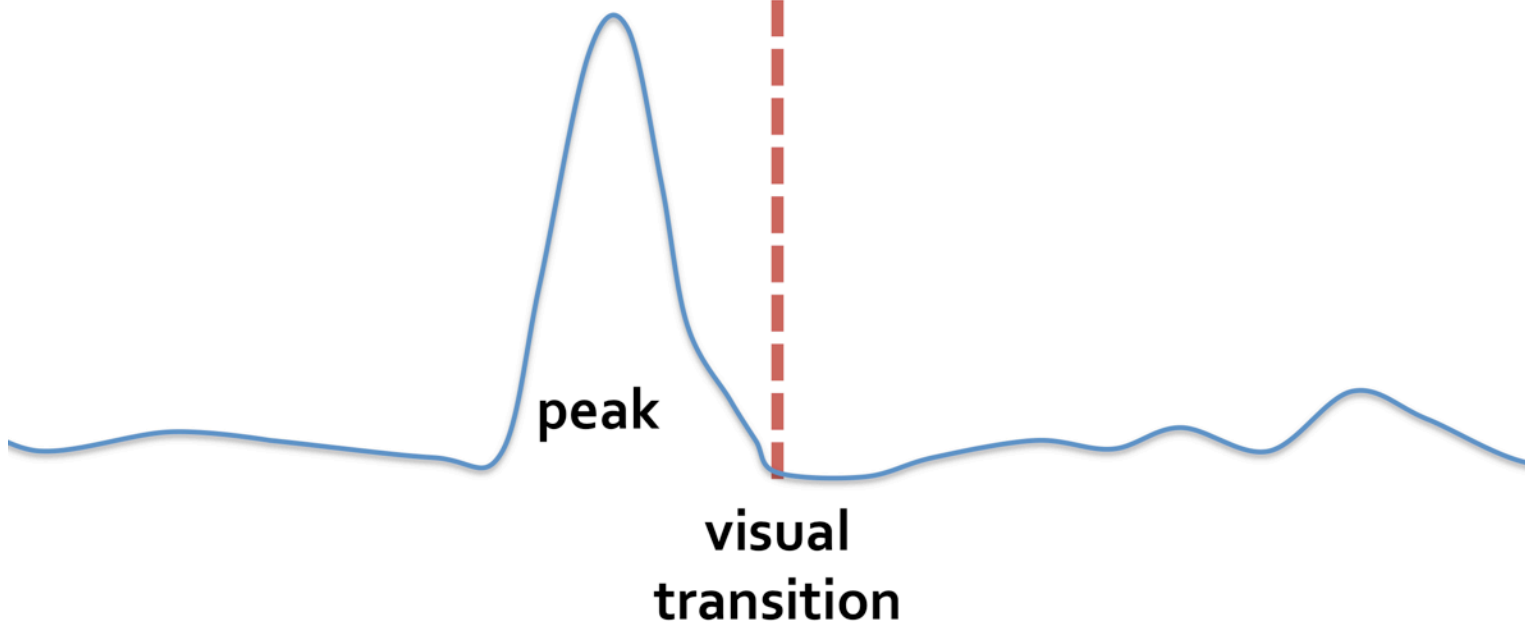
# Example: Backing up

before transition

```
def fact(n):
    """assumes that n is an int > 0
    returns n!"""
    res = 1
    while n > 1:
        res = res * n
        n = n - 1
    return res
```

*Update*

after transition



# LectureScape: Enhancing lecture videos



The interface displays a video player for a lecture titled "Recursion on Strings". The video content shows a slide titled "How to solve this recursively?" with the following bullet points:

- First, convert the string to just characters, by stripping out punctuation, and converting upper case to lower case
- Then
  - Base case: a string of length 0 or 1 is a palindrome

The slide also features a word cloud with terms like "palindrome", "characters", "string", "left", "onward", "drum", "ispalindrome", and "lowercase".

Annotations on the left side of the interface include:

- Word Cloud**: Points to the word cloud on the slide.
- Lecture Video**: Points to the video player area.
- Interaction Peaks**: Points to the peaks on the rollercoaster timeline.
- Rollercoaster Timeline**: Points to the timeline at the bottom of the video player.

Annotations on the right side of the interface include:

- Keyword Search**: Points to the search bar at the top right.
- Interactive Transcript**: Points to the transcript on the right side.

The transcript on the right side contains the following text:

So how do we solve it?  
We first convert the string to just characters. We'll look to that in a second.  
And solving it recursively is actually pretty easy. **If I have a string that's either of length zero or of length one, it's a palindrome.**  
So length one is just one character.  
Otherwise, to solve this, what I'm going to do is take the string and ask the following question.  
If the first and last character are the same, then they satisfy the condition.  
And let me then simply look at the remaining string, throwing away the first and last character, and ask is that a palindrome.  
Wonderful.  
There's that recursive property again.  
If I can break it down into that problem, I'm set. So I could write code to do that.  
Just to give you the example again, this says I'm going to take something like "Able was I were I saw Elba" and reduce it to just that string of characters without the spaces or any punctuation. And then, to test whether that string is a palindrome, that's the same as asking are the first and last characters the same?

The video player controls at the bottom show a play button, a progress bar at 1:44 / 7:29, a volume icon, and an "Add Bookmark" button. A tooltip indicates that the user watched a segment from 4:06 to 4:25.



# SOLUTION ANALYTICS



**Elena  
Glassman**



# A Typical Programming Assignment

Time



Write an iterative function that computes  $a^b$



What did our students do?

Oh no! Never do that!

Oops, the autograder should have caught that.

Clever--I didn't know you could do it that way.

- Overcode allows teaching staff to see the similarity and variation among thousands of solutions.

iterPower

done

showing 862 total stacks  
 that 3842 total  
 represent submissions

filtering by:

nothing yet

1534

id: 1

```
def iterPower(base,exp):
    result=1
    while exp>0:
        result*=base
        exp-=1
    return result
```

374

id: 3

```
def iterPower(base,exp):
    result=1
    while exp>0:
        result=result*base
        exp-=1
    return result
```

153

id: 727

```
def iterPower(base,exp):
    result=1
    while exp>0:
```

Filter

Rewrite

Legend

lines that appear in at least

50

submissions

77

base=resultB

2592

def iterPower(base,exp):

701

def iterPower(base,expB):

349

def iterPower(base,expC):

51

def iterPower(base,expD):

51

def iterPower(resultB,expC):

55

elif expC==1:

527

else:

2466

exp-=1

279

exp=exp-1

135

exp=expB

366

expC-=1

65

expC=expC-1

63


for i in range(0,expB):

174

for i in range(expB):

52

iC+=1

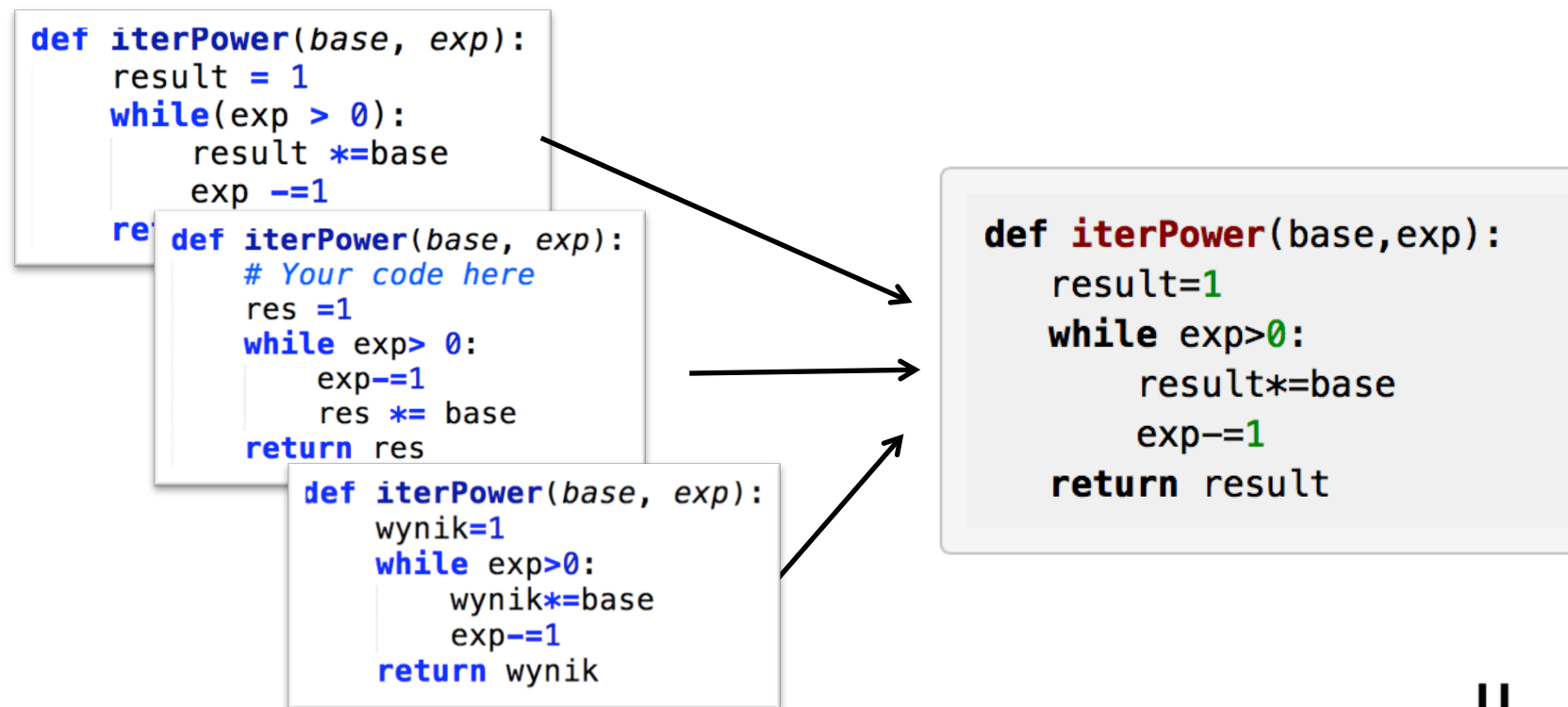


14

MIT HUMAN-COMPUTER INTERACTION

# Solution Cleaning & Clustering

- OverCode makes solutions easier to read and cluster
  - Reformat code for consistency
  - Rename variables with identical behavior
  - Ignore statement order when clustering solutions



Total Solutions

Largest Stack

2nd Largest

iterPower

~3800

1534

```
def iterPower(base,exp):
    result=1
    while exp>0:
        result*=base
        exp-=1
    return result
```

~40%

374

```
def iterPower(base,exp):
    result=1
    while exp>0:
        result=result*base
        exp-=1
    return result
```

~10%

hangman

~1100

97

```
def getGuessedWord(secretWord,
    result='')
    for letter in secretWord:
        if letter in lettersGue
            result+=letter
        else:
            result+='_'
    return result
```

~9%

92

```
def getGuessedWord(secret
    resultB='')
    for letter in secretW
        if letter in lette
            resultB+=lette
        else:
            resultB+='_'
    return resultB
```

~9%

computeDeriv

~1400

22

```
def computeDeriv(poly):
    result=[]
    if len(poly)==1:
        return[0.0]
    for i in range(1,len(poly))
        result.append(float(pol
    return result
```

~1%

13

```
def computeDeriv(poly):
    result=[]
    if len(poly)<2:
        return[0.0]
    for i in xrange(1,len
        result.append(floa
    return result
```

~0.5%

# Performance

OverCode preprocessing pipeline is **linear** with number of solutions and runs on a **laptop**

Problem	Correct Solutions	Running Time	Initial Stacks	Common Variables
<code>iterPower</code>	3875	15m 28s	862	38
<code>hangman</code>	1118	8m 6s	552	106
<code>compDeriv</code>	1433	10m 20s	1109	50

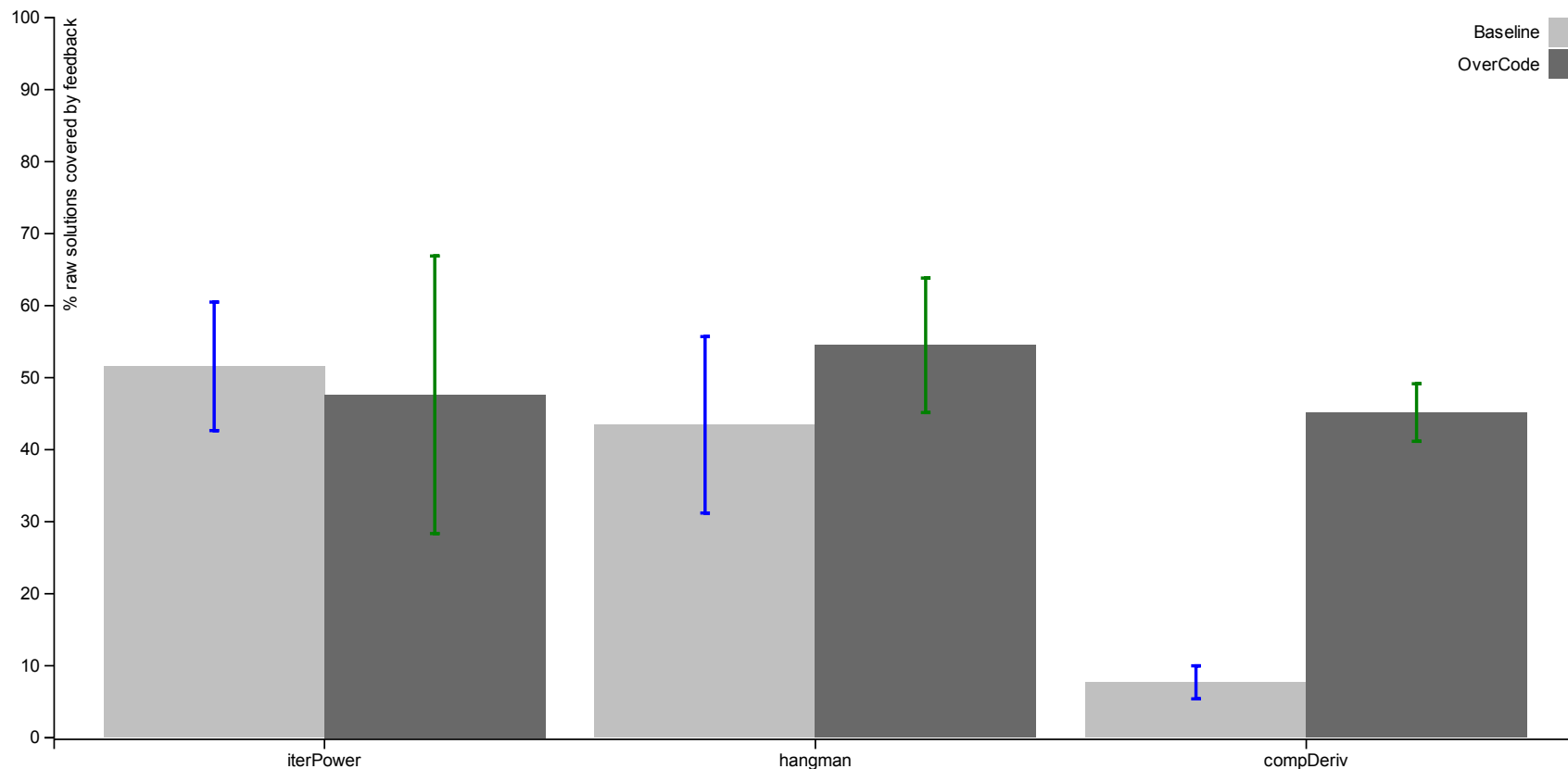
Other clustering approaches are quadratic in number of solutions and need a computer cluster.



# Feedback Coverage

% solutions covered  
by the teacher's post

users: 12 teaching assistants  
control: all solutions concatenated in a page  
task: write a discussion forum post

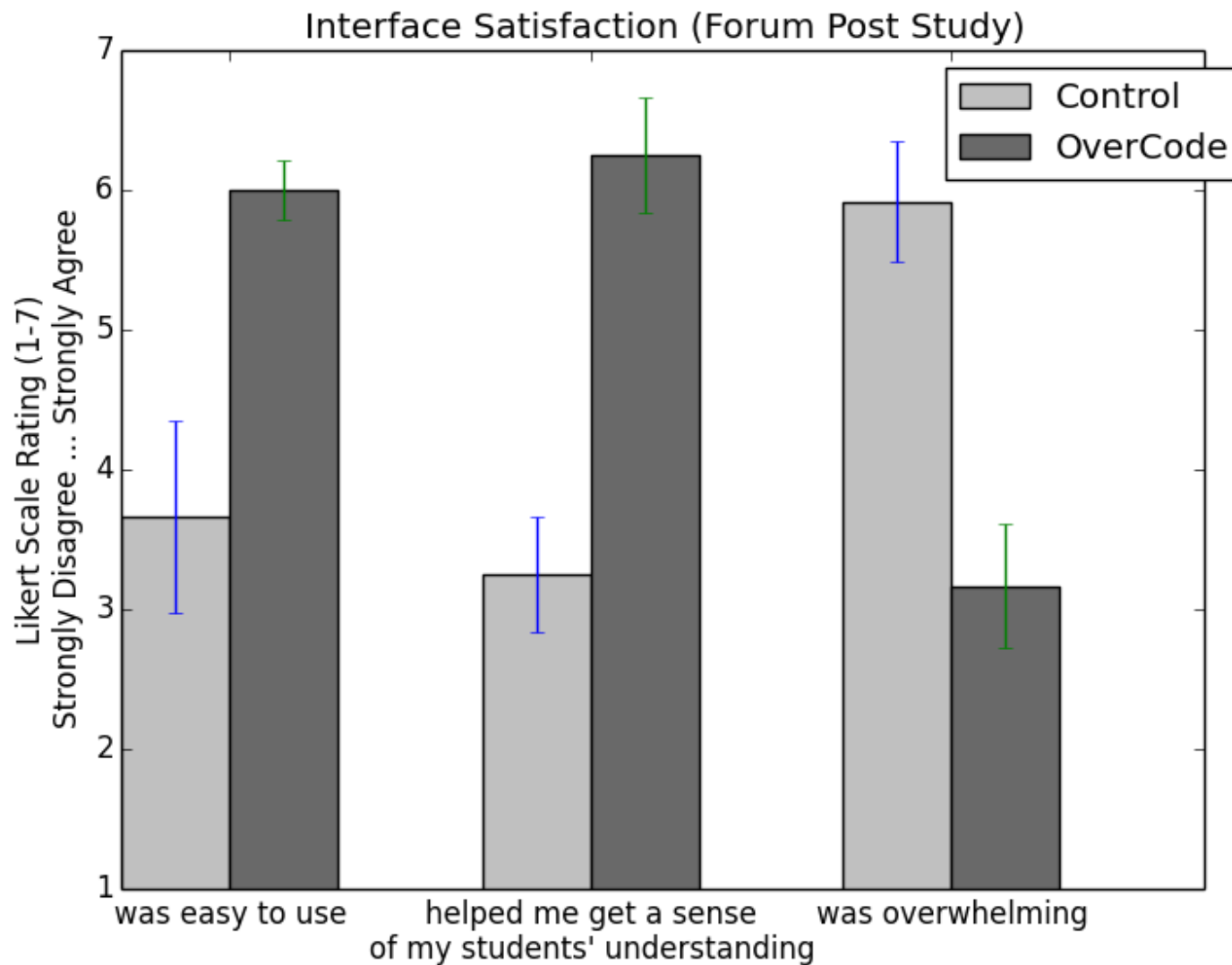


iterPower

hangman

computeDeriv

# Teacher Perceptions



# PEER CODE REVIEW

# Problem: Feedback about Coding Style

- MIT 6.005 Software Construction
  - foundation-level programming course (replaced 6.001/6.170)
  - 400 students per year, mostly sophomores
- Students write lots of code
  - roughly 10kloc in problem sets and projects
- Automatic grading is necessary but not sufficient

```
// compute n! requires n >= 0
int factorial(int n) {
    if (n == 0) return 1;
    else return n * factorial(n-1);
}
```

correct and understandable

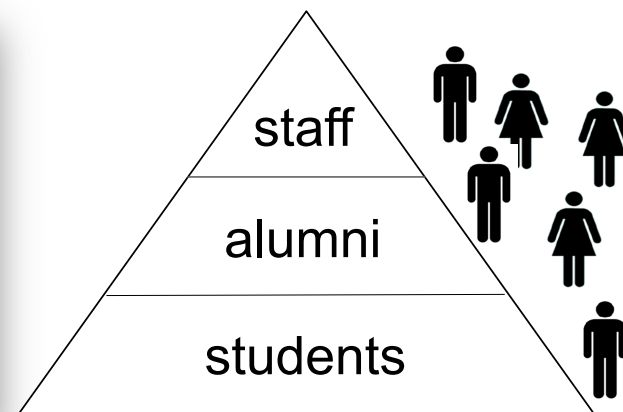
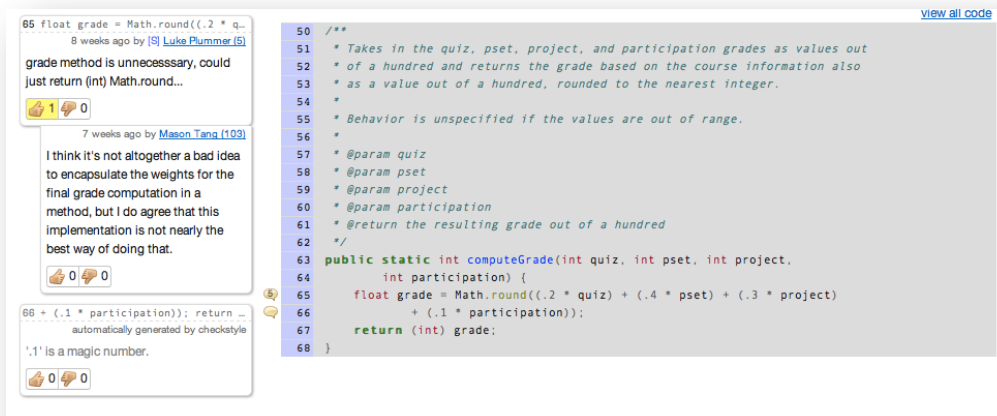
```
int factorial(int n) {
    int i, result=1;
    if (n == 0) result = 1;
    else {
        for (i = 1; i < n; ++i) result *= i;
        result = result*n;
        return result;
    }
    return 1;
}
```

correct but confusing

- we need human readers, and we want line-by-line feedback

# Approach: Crowd-Driven Code Review

- Chop up student programs into **chunks**
- Review the chunks by a **mixed crowd**: students, staff, alums



- Anticipated benefits
  - faster, cheaper, more diverse comments
  - give practice with code reviewing (a widespread industry practice)
  - expose to good and bad solutions
  - reduce workload on teaching staff
  - incorporate alumni back into the course
- Not using for grading... yet



# Caesar: Divide & Conquer

programs chopped into **chunks**

each chunk assigned to multiple reviewers

```

14 public class RulesOf6005 {
15
16
17 /**
18  * Tests if the string is one of the items in the Course Elements section.
19  *
20  * @param name - the element to be tested
21  * @return true if <name> appears in bold in Course Elements section. Ignores case (capitalization).
22  * Example: "Lectures" and "lectures" will both return true.
23  */
24 public static boolean hasFeature(String name){
25     // TODO: Fill in this method, then remove the RuntimeException
26     String[] elements = { "lectures", "recitations", "laptops required", "text", "problem sets", "i
27     String test = name.toLowerCase();
28     for (int ii = 0; ii < 9; ii++) {
29         if (elements[ii].equals(test)) {
30             return true;
31         }
32     }
33     return false;
34 }
35
36
37 /**
38  * Takes in the quiz, pset, project, and participation grades as values out of a
39  * hundred and returns the grade based on the course information also as a value out
40  * of a hundred, rounded to the nearest integer.
41  *
42  * Behavior is unspecified if the values are out of range.
43  *
44  * @param quiz
45  * @param pset
46  * @param project
47  * @param participation
48  * @return the resulting grade out of a hundred
49  */
50 public static int computeGrade(int quiz, int pset, int project, int participation){
51     return (int)Math.round((quiz*.2) + (pset*.4) + (project*.3) + (participation*.1));
52 }
53
54
55 /**
56  * Based on the slack day policy, returns a date of when the assignment would be due, making sure not
57  * exceed the budget. In the case of the request being more than what's allowed, the latest possible
58  * due date is returned.
59  *
60  * Hint: Take a look at http://download.oracle.com/javase/6/docs/api/java/util/GregorianCalendar.html
61  *
62  * Behavior is unspecified if request is negative or duedate is null.

```

## code to review

PrimeFactorsServer	5	1
PrimeFactorsServer	5	1
EchoClient	1	1
EchoClient	5	1
EchoClient	3	1
EchoServer	3	1
PrimeFactorsClient	6	1
PrimeFactorsServer	6	1
EchoClient	2	1
EchoClient	2	1

## code recently reviewed

RulesOf6005.extendDeadline(..)	18	2
RulesOf6005.extendDeadline(..)	4	3
RulesOf6005.computeGrade(..)	9	3
RulesOf6005.extendDeadline(..)	6	2
RulesOf6005.computeGrade(..)	6	3
RulesOf6005.hasFeature(..)	3	2
RulesOf6005.hasFeature(..)	6	2
RulesOf6005.hasFeature(..)	4	2
RulesOf6005.hasFeature(..)	5	2
RulesOf6005.hasFeature(..)	4	2

# Social Reviewing

automatic  
style checker  
comments

```
12 /** * @param expression * a Stri...
    automatically generated by checkstyle
    file contains tab characters (this is the
    first instance).
```

👍 1 👎 0

reviewer  
comments

```
13 * @param expression * a String r...
    5 weeks ago by Jason Juang \(104\)
```

Specify here that expression must not be null. You're throwing an unchecked exception when that happens, so you owe it to whoever is calling your method to explain that you're going to crash their program if they do that.

👍 1 👎 1

upvotes &  
downvotes

replies &  
discussion

4 weeks ago by [T] [Robert C Miller \(107\)](#)

But a NullPointerException is the typical result when passing null anyway, and that's unchecked too.

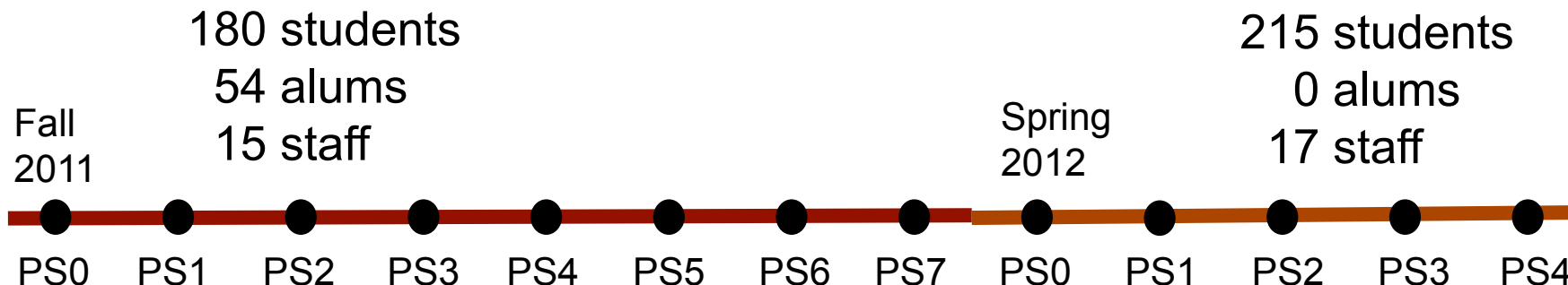
[view all code](#)

```
12 /**
13  * @param expression
14  *      a Str
15  *      the p
16  * @return the valu
17  *      units, e
18  */
19 public String evaluate(String
20     if (expression == null) {
21
22     Lexer lexer = new Lexer(ex
23     Parser p = new Parser(lexe
24     return p.evaluate().toStri
25 }
```

reviewers can see  
whole program  
(not just chunk)  
if needed

reviewers have a  
reputation (#upvotes,  
+ 100 if they're alums  
or staff of the course)

# Experience



13 problem sets, 2200 submissions

21,500 comments

5% alums

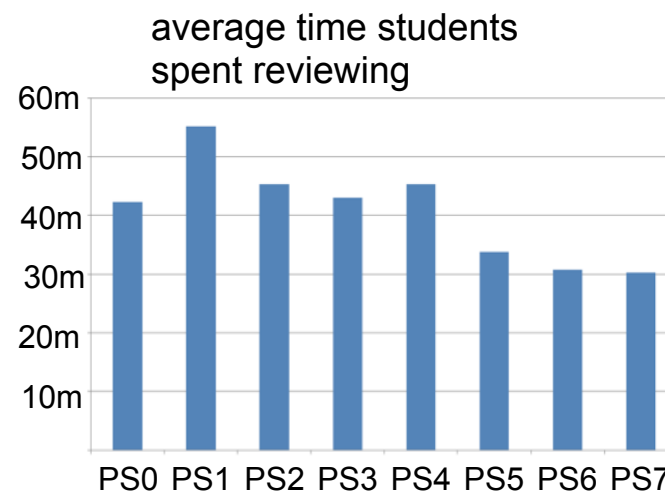
8% staff

87% students

16.2% upvoted

0.7% downvoted

9.6 comments per submission



# Kinds of Comments

Bug  
Clarity  
Performance  
Simplicity  
Style  
Positive  
Learning

- I don't really understand what you're
- This is nice and concise. (I didn't know you could iterate through an array like this in a for loop)
- This is interesting. Why do you store all the messages you send/receive in a log?

**Code author:** For debugging. The log adds time stamps, which help a lot for debugging concurrency problems.

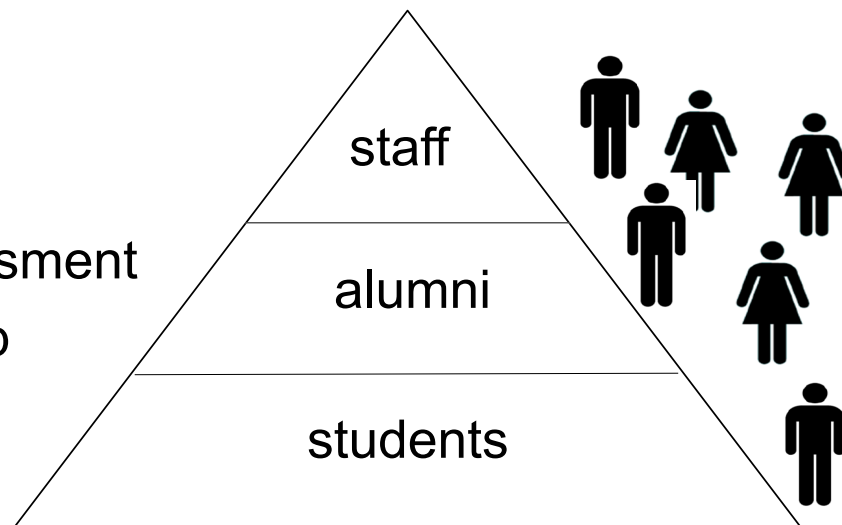


# A LOOK AHEAD

# MOOCs Have to Run Themselves

- Launching a MOOC is like authoring a textbook
  - But keeping it running currently requires sustained expert involvement
  - In the long run, we can teach the world for free only if we don't have to staff the MOOCs

- Implications
  - Intelligent tutor systems
  - Peer help, feedback, assessment
  - Alumni or external staff help



# MOOCs Have to Improve Themselves

---



- edX and Coursera will be littered with stale MOOCs
  - Because faculty have no time or incentive to revise them
  - In the long run, MOOCs have to revise and improve themselves, automatically
- Implications
  - Crowdsourced content: exercises, quizzes, textbook, videos
  - FrankenMOOCs that combine the best stuff out there
  - Video content that can be edited like Wikipedia



# MOOCs Are Big Data for Education

---

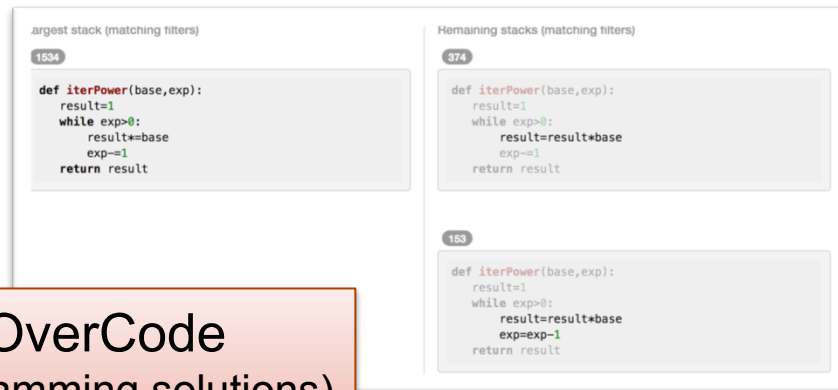
- Google and Bing drive information retrieval research
  - because they own the data & control the interface
- Facebook and Twitter increasingly drive social network research
  - again: data + interface
- Universities could be driving learning science in CS
  - if we step up and take ownership of the data + the interface



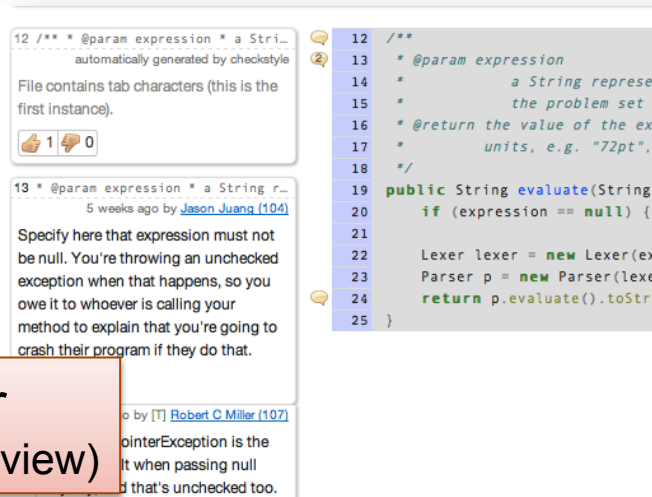
# Summary



Lecturescape  
(lecture videos)



OverCode  
(programming solutions)



Caesar  
(peer code review)

future

- self-running MOOCs
- self-improving MOOCs
- MOOCs are our big data

Thanks to support from NSF, Quanta Computer, Google, edX