# Crowd Production, Peer Production

CS 278 | Stanford University | Michael Bernstein

# Last time

Crowdsourcing: an open call to a large group of people who self-select to participate

Crowds can be surprisingly intelligent, if opinions are levied with some expertise and without communication, then aggregated intelligently.

Design differently for intrinsically and extrinsically motivated crowds

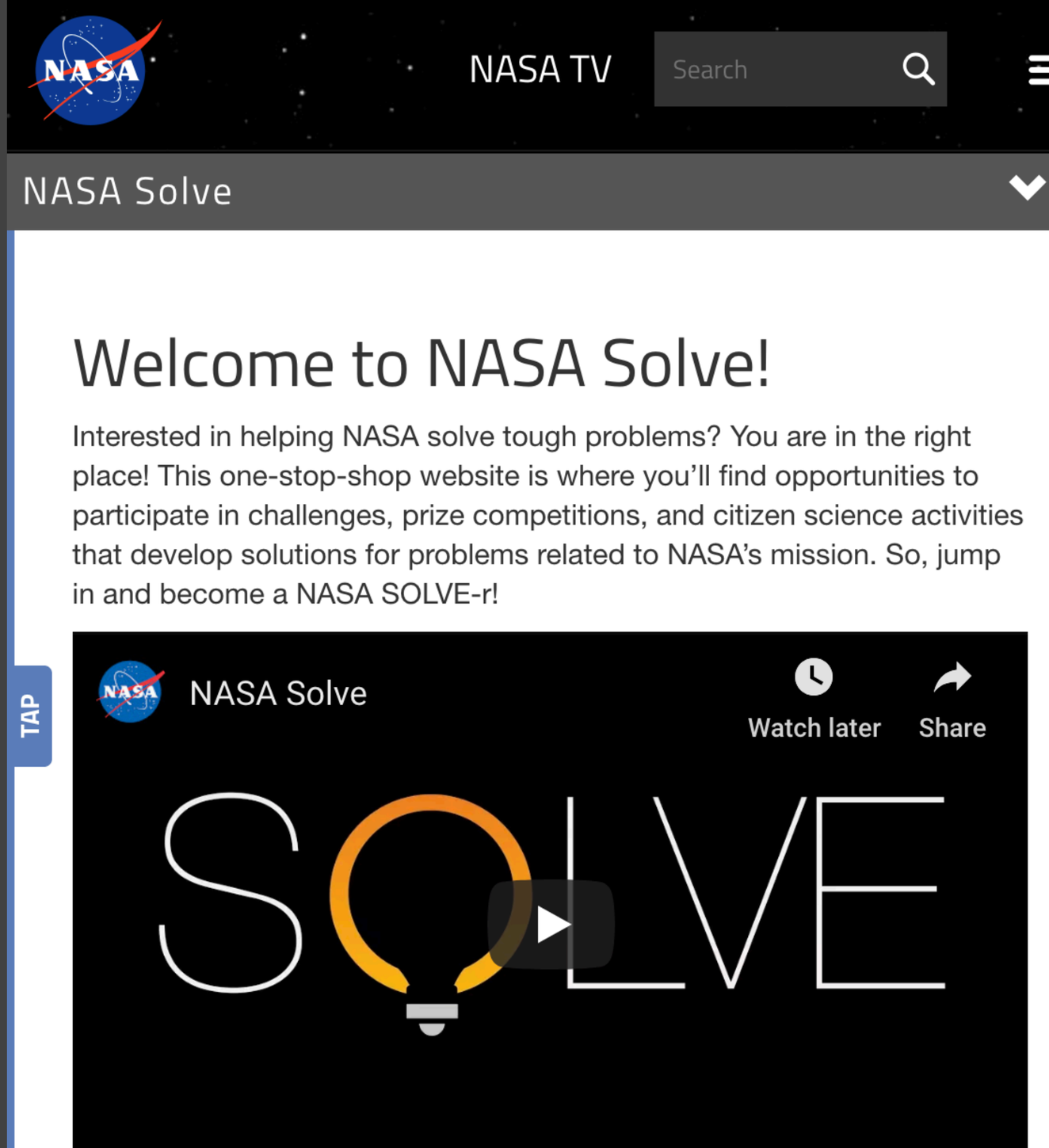
Quality issues are best handled up front by identifying the strong contributors and gating them through

# Last time

Parallel, independent contributions

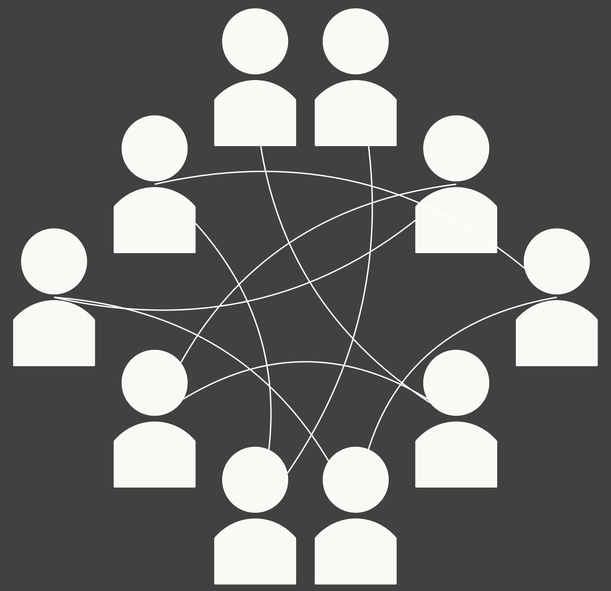
But, this only works if the goal can be subdivided into modular components with few or no interdependencies.

Think filling out rows of a spreadsheet or taking argmax



# Today

Interdependent, integrated contributions



Think invention, engineering, or game design.

# How?

There are fundamental differences between parallel and interdependent contribution structures.

We can’t just make a movie or build Linux with parallel contributions.

Johnny Cash Project: crowdsourced music video
One frame per participant — beautiful, slightly anarchic

Star Wars Uncut: crowdsourced movie remake, 2hr long
One scene per participant — style whiplash

# How?

There are fundamental differences between parallel and interdependent contributions. We can't just make a movie or build Linux with parallel contributions.

So, how do we create complex outcomes with distributed online collaborations?

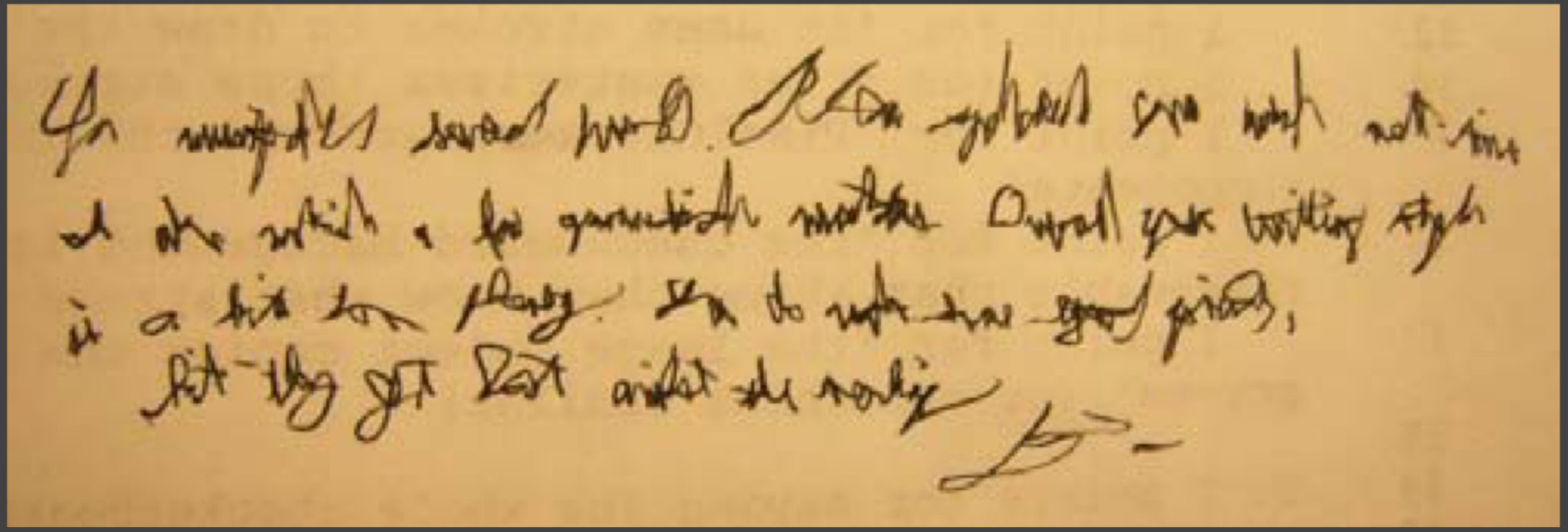
Topics:

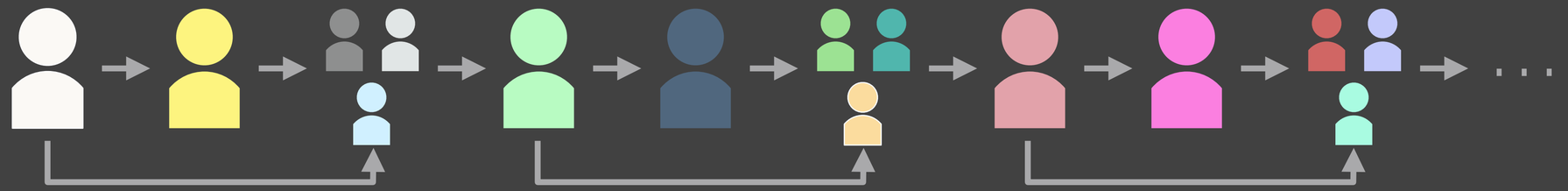
- Workflows
- Peer production
- Convergence and coordinated adaptation

# Workflows

# Iterative crowd algorithm

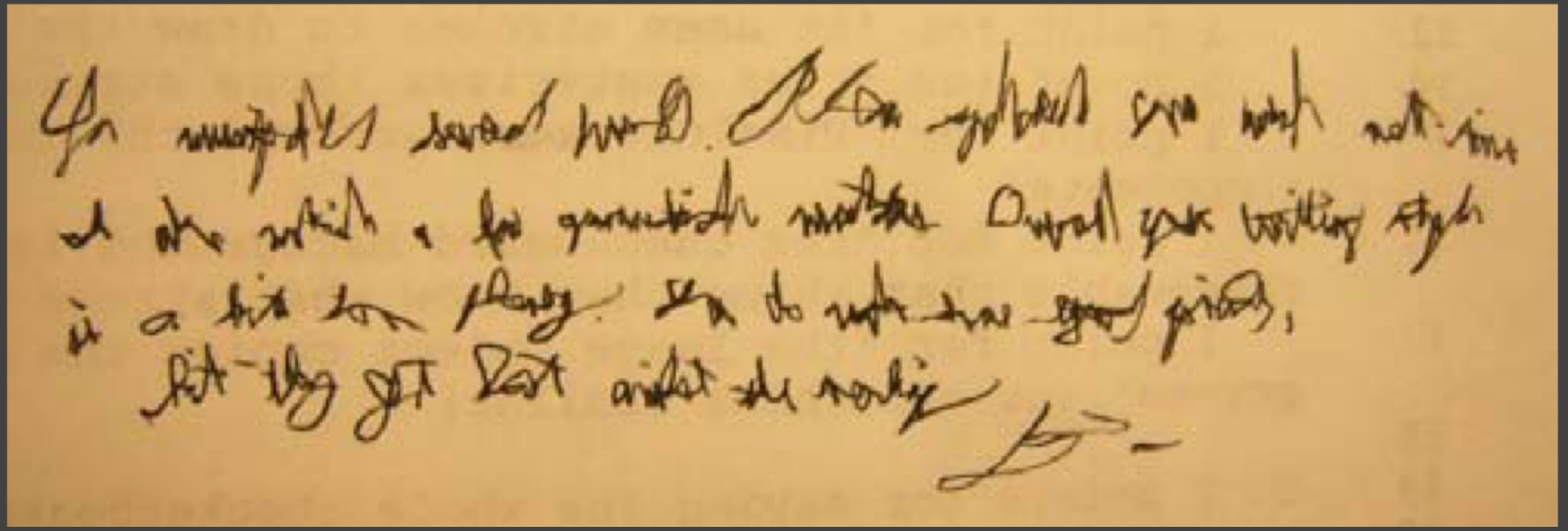
[Little et al. 2009]



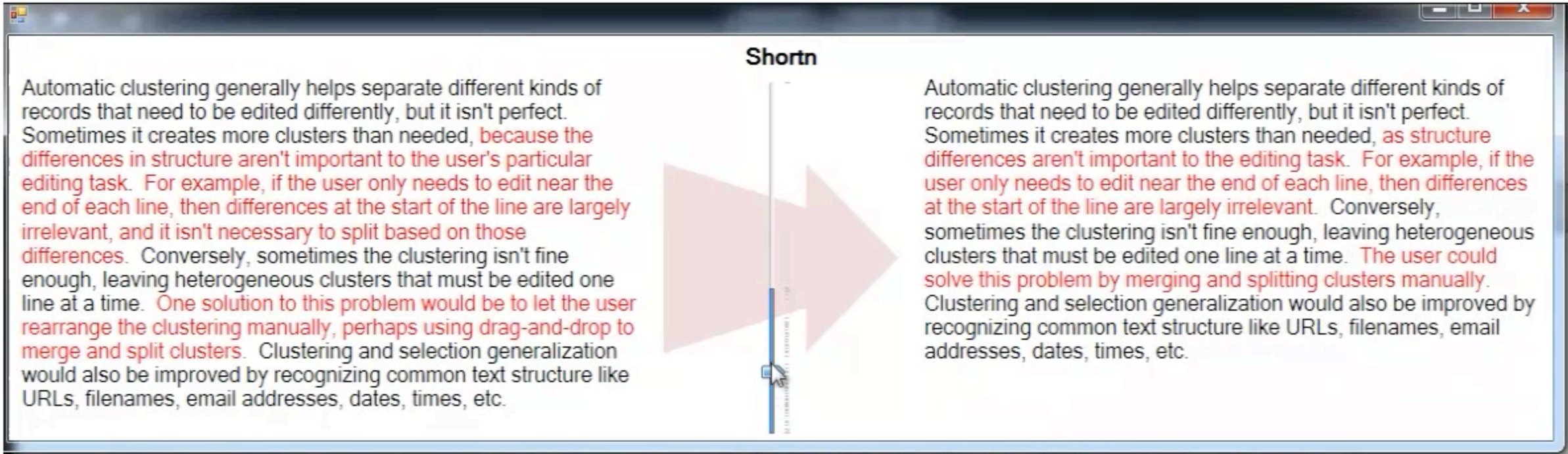


# Iterative crowd algorithm

[Little et al. 2009]



You (misspelled) (several) (words). Please spellcheck your work next time. I also notice a few grammatical mistakes. Overall your writing style is a bit too phoney. You do make some good (points), but they got lost amidst the (writing). (signature)



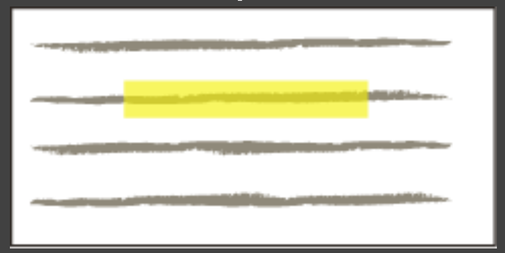

Automatic clustering generally helps separate different kinds of records that need to be edited differently, but it isn't perfect. Sometimes it creates more clusters than needed, because the differences in structure aren't relevant to a specific task. For example, if the user only needs to edit near the end of each line, then differences at the start of the line are largely irrelevant, and it isn't necessary to split based on those differences. For example, if the user only needs to edit near the end of each line, then differences at the start of the line are largely irrelevant. Conversely, sometimes the clustering isn't fine enough, leaving heterogeneous clusters that must be edited one line at a time. The user could solve this problem by merging and splitting clusters manually. Clustering and selection generalization would also be improved by recognizing common text structure like URLs, filenames, email addresses, dates, times,

# Find-Fix-Verify

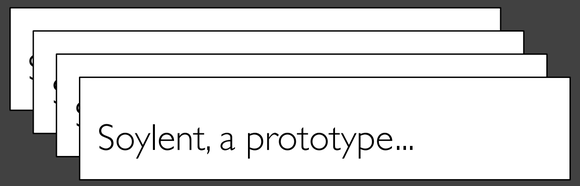
[Bernstein et al. 2010]

Find-Fix-Verify is a design pattern for open-ended tasks.

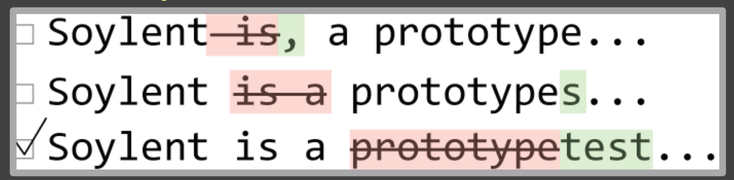
## Find a problem



## Fix the problem

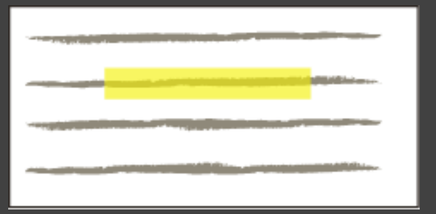



## Verify each fix



**Find**

"Identify at least one area that can be shortened without changing the meaning of the paragraph."



Independent agreement to identify patches

**Fix**

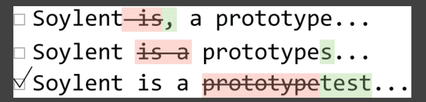
"Edit the highlighted section to shorten its length without changing the meaning of the paragraph."

Soylent, a prototype...

Randomize order of suggestions

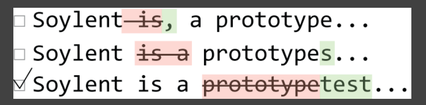
**Verify**

"Choose at least one rewrite that has style errors, and at least one rewrite that changes the meaning of the sentence."

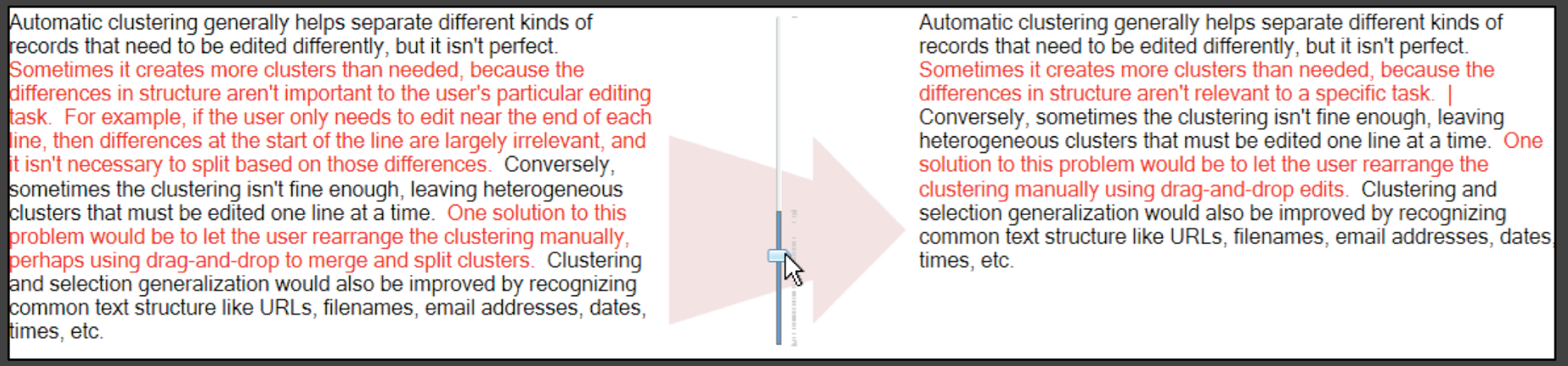


# Verify

"Choose at least one rewrite that has style errors, and at least one rewrite that changes the meaning of the sentence."



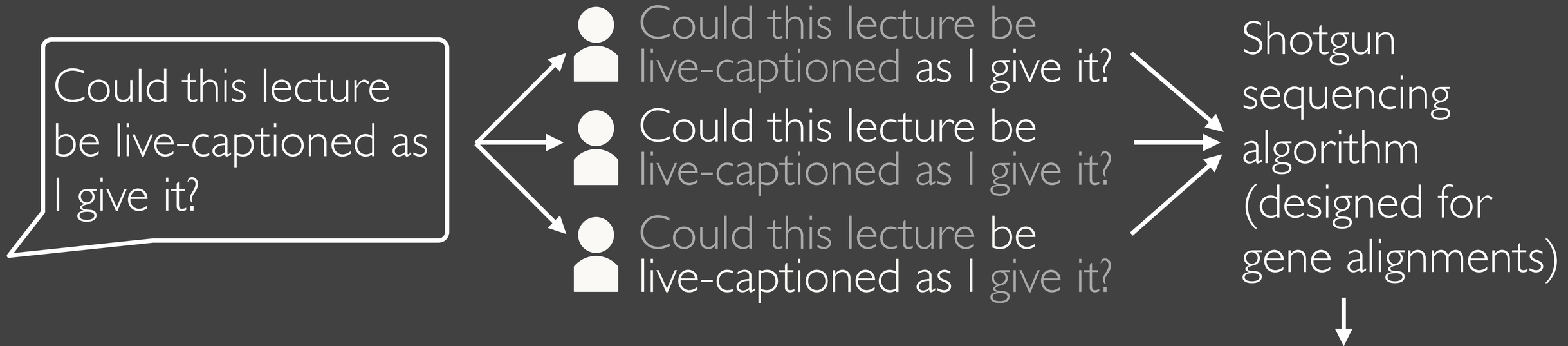

Keep suggestions that do not get voted out



# Realtime crowdsourcing

[Lasecki et al. 2012]

Can crowds achieve real-time responses?



# Crowds of experts

Crowd workers

microtask worker
microtask worker
microtask worker
microtask worker
microtask worker



Experts

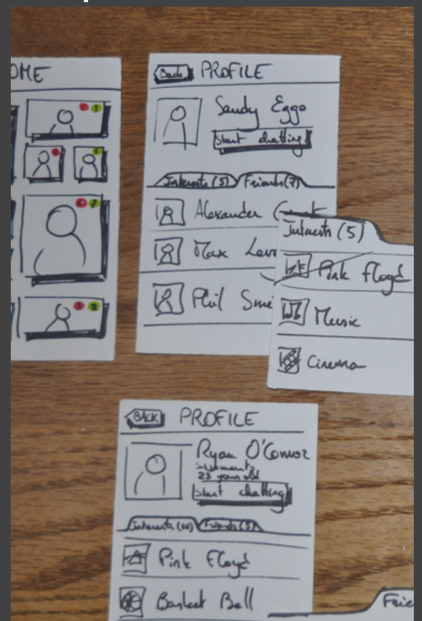
programmer
designer
video editor
musician
statistician

# Flash Teams

[Retelny et al., UIST '14]

Computationally-guided teams of crowd experts supported by lightweight team structures.

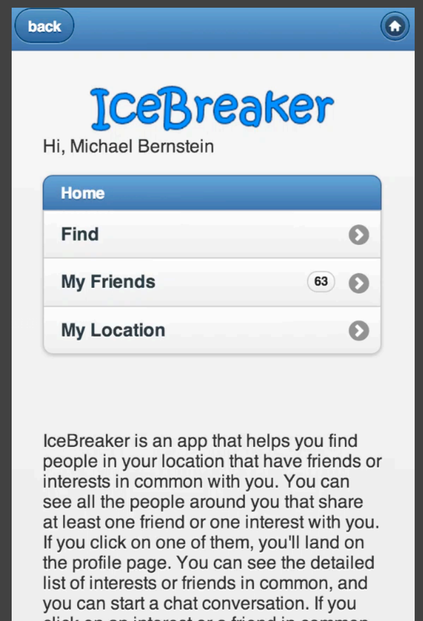
Input




Flash Team

Design workflow

Output



# Future of work

Crowdsourcing is a populist form of information work, but the technical infrastructure actively disempowers workers. [Irani and Silberman '13]

How do we design a future workplace that we want our children to join? [Kittur et al. '12]

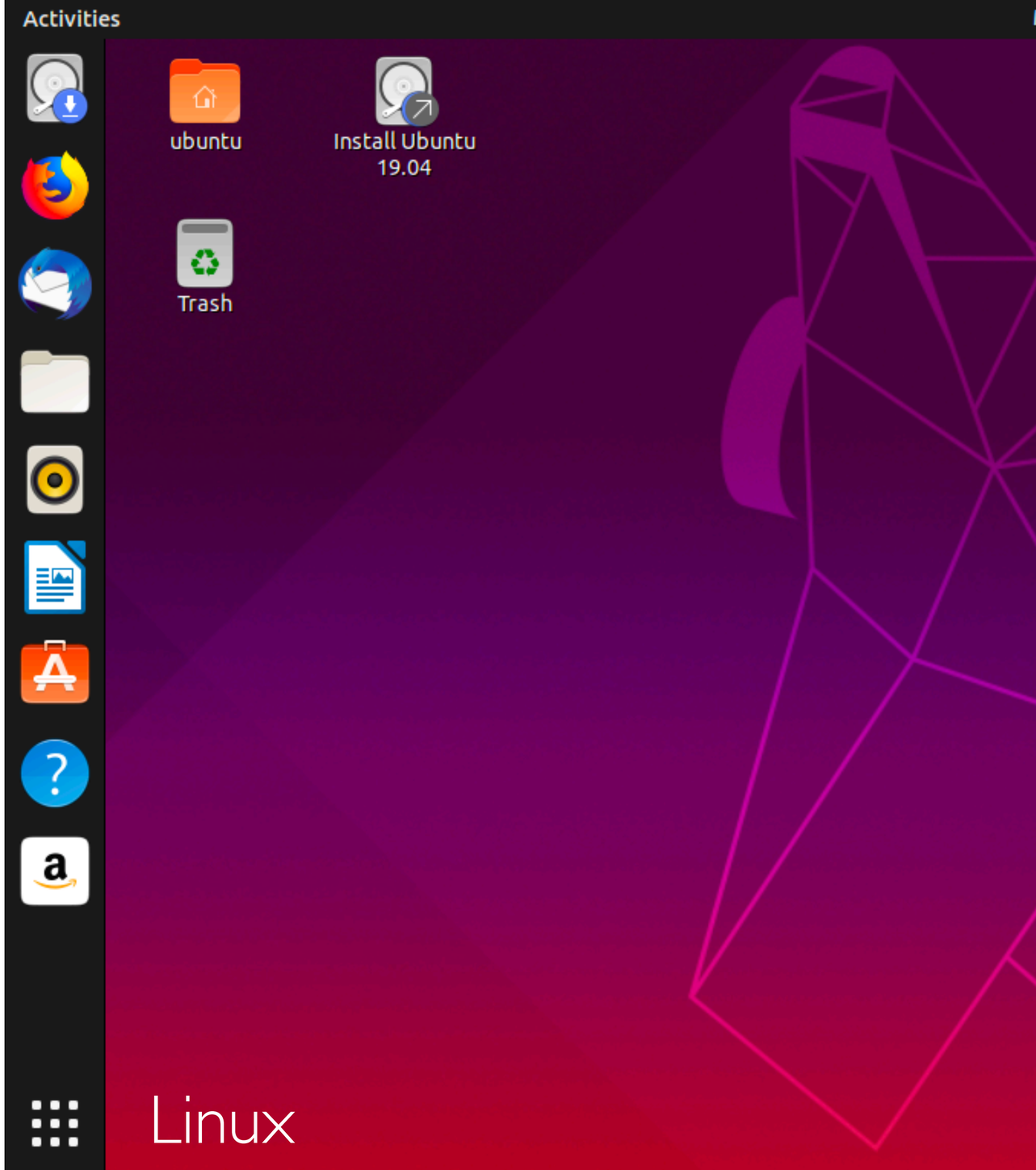
One shorthand thought keep in mind: autonomy. And for whose benefit are these workflows?

More on this to come.

# Peer production

# Wikipedia

The Free Encyclopedia



# What is peer production?

Crowdsourcing: making an open call to a large set of individuals who self-select into tasks

Peer production includes additional requirements… [Benkler 2009]

- Decentralized conception: many control the direction and outcome, not a traditional bureaucracy
- Diverse motivations: especially non-monetary incentives
- Results treated as a commons: the output is publicly available and generally non-rival (def: when I use it, it doesn't reduce your ability to use it)
- No contracts: governance and work allocation isn't handled through signed contracts

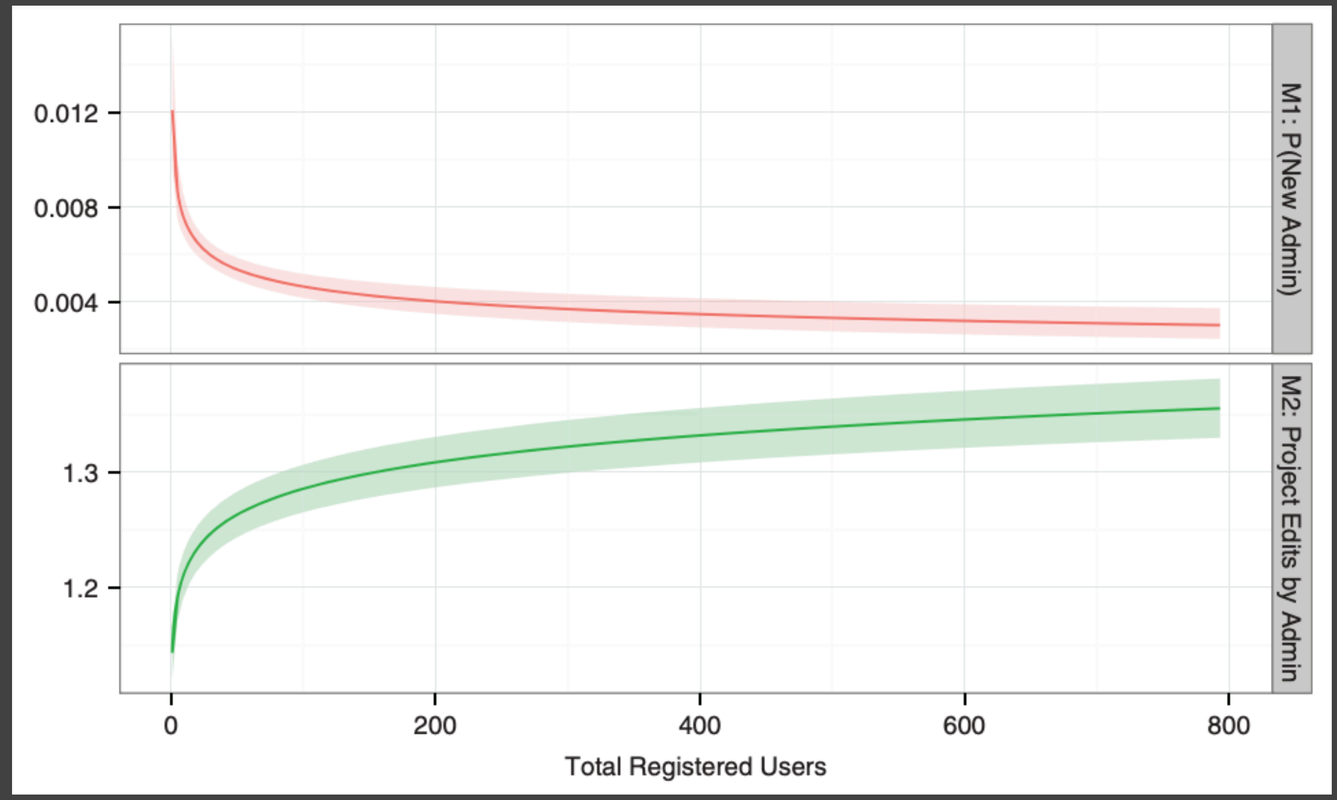
# When does peer production work?

Benkler's argument [2002] is that peer production outperforms traditional firms when there exists strong intrinsic motivation and work can be broken down into granular and easy-to-integrate tasks.

# What role does leadership play in peer production?

While open-source projects and collaborative wikis sound very decentralized, in practice, leadership hierarchies emerge. [Benkler, Shaw and Hill 2016]

As a system grows, it's harder to become an admin [Shaw & Hill 2014]



# Governance models

[https://opensource.guide/leadership-and-governance]

BDFL: "Benevolent Dictator for Life" who makes all final decisions.

Examples: Ethereum, Django, Swift, Ruby, Pandas, Ubuntu, Linux, SciPy, Perl

Meritocracy: top contributors are granted decision-making rights. Policy decisions via committee vote.

Examples: Red Hat, all Apache projects

Issue: outspoken people get credit, disempowering many communities

Liberal contribution: allow as many contributors as possible, and use consensus-seeking for policy decisions

Examples: node.js and Rust

# Convergence and coordinated adaptation

# Limits of algorithmic coordination

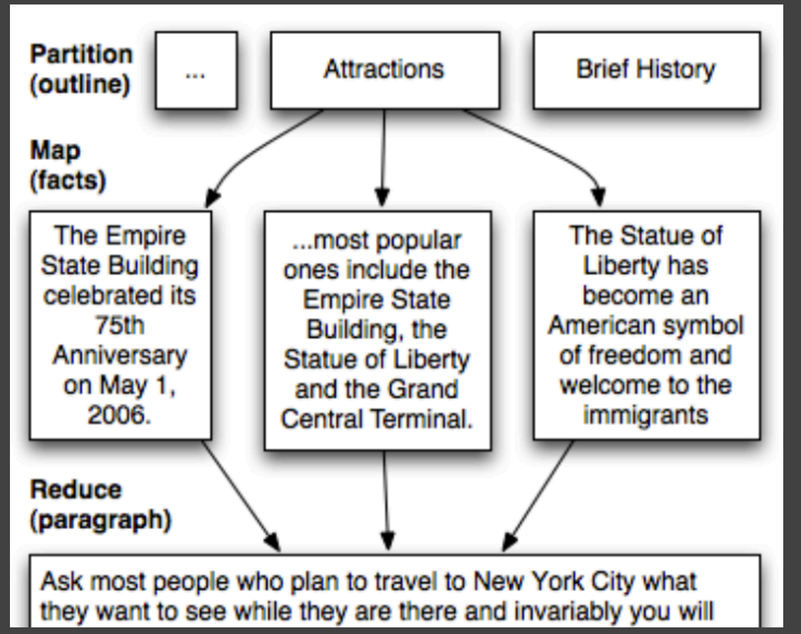
So far, goals such as invention, production, and engineering have remained largely out of reach [Kittur et al. 2013]

Why?

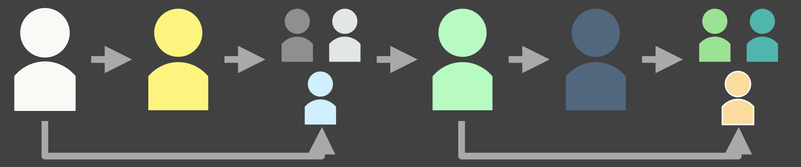
# Dominant architecture: algorithms

Modularize and pre-define all possible behaviors into workflows

Computation decides which behaviors are taken, when, and by whom; optimizes, error-checks, and combines submissions




[Kittur 2011]



[Little 2010]
[Dai and Weld 2010]

# Limits of algorithmic coordination

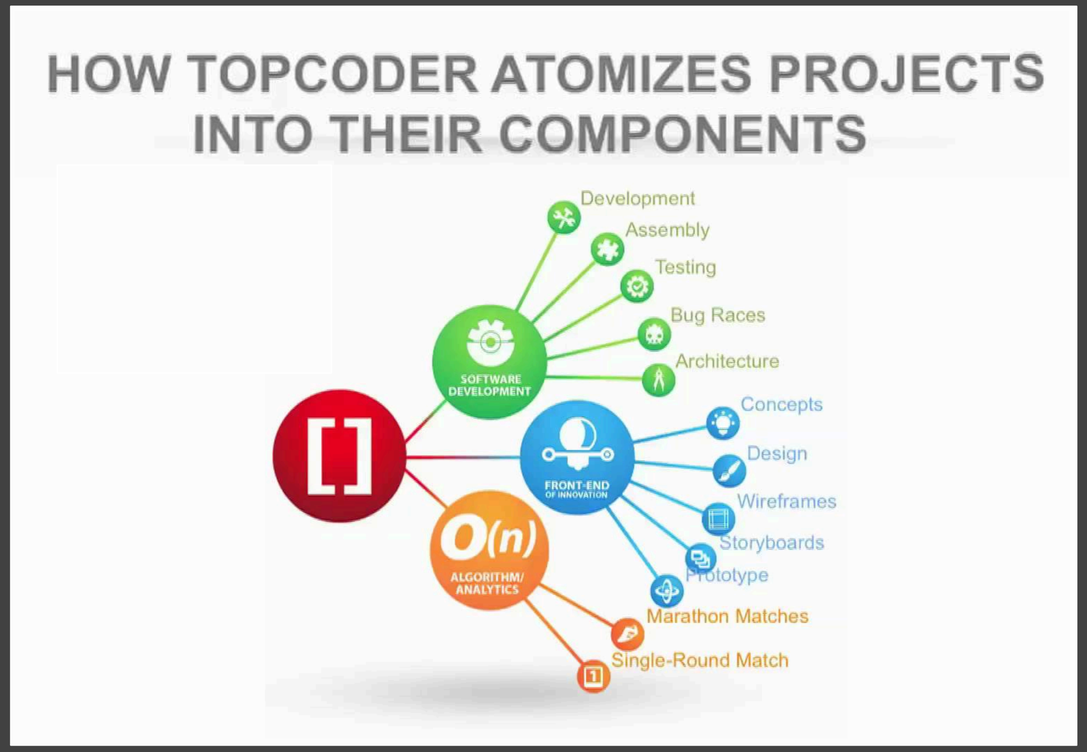
Returning to the question: why have complex goals remained largely out of reach?

Open-ended, complex goals are fundamentally incompatible with a requirement to modularize and pre-define every behavior [Van de Ven, Delbecq, and Koenig 1976; Rittel and Weber 1973; Schön 1984]

# Limits of crowdsourcing and peer production

"Peer production is limited not by the total cost or complexity of a project, but by its modularity." [Benkler 2002]

"With the Linux kernel […] we want to have a system which is as modular as possible. The open–source development model really requires this, because otherwise you can't easily have people working in parallel." [Torvalds 1999]




[Boudreau, Lacetera, and Lakhani 2011]

# Interdependence and collective action remain challenging

The result: algorithmic, workflow-based architecture confines collaborations to goals so predictable that they can be entirely modularized and pre-defined.

But many valuable collective activities do not fit this criteria.

BLACK LIVES MATTER


Tesla construction

Credit: @elonmusk on Twitter


UN climate change meeting

Credit: UNClimateChange on Flickr

# Why are these challenging?

Convergence: crowds are excellent at generating ideas and at spreading awareness, but it's much more challenging for them to build consensus toward a single action.

(This was noted as a challenge that the Occupy movement faced.)

# Convergence [Example via Niloufar Salehi]

The New York Times

Twitter Users Split on Boycott Over Platform's Move Against Rose McGowan

VOGUE

CULTURE > OPINION

Why I'm Not Joining the Women Boycotting Twitter Today

The Washington Post

#WomenBoycottTwitter revives post-election conversations about the lack of solidarity among women

MADAMENOIRE

Black Women Commandeer #WomenBoycottTwitter And Turn It Into A Celebration Of Ourselves With #WOCAffirmation

# Convergence




[Example via Niloufar Salehi]

# Why are these challenging?

Coordinated adaptation: changing direction in sync with each other.

Crowds are excellent at executing pre-defined tasks, but it's much more challenging for them to continually re-evaluate goals and adapt in sync.

# Hybrid peer production

Why is it that many successful peer production projects form traditional organizations to support their efforts?

MongoDB: MongoDB, Inc.

Ubuntu: Canonical

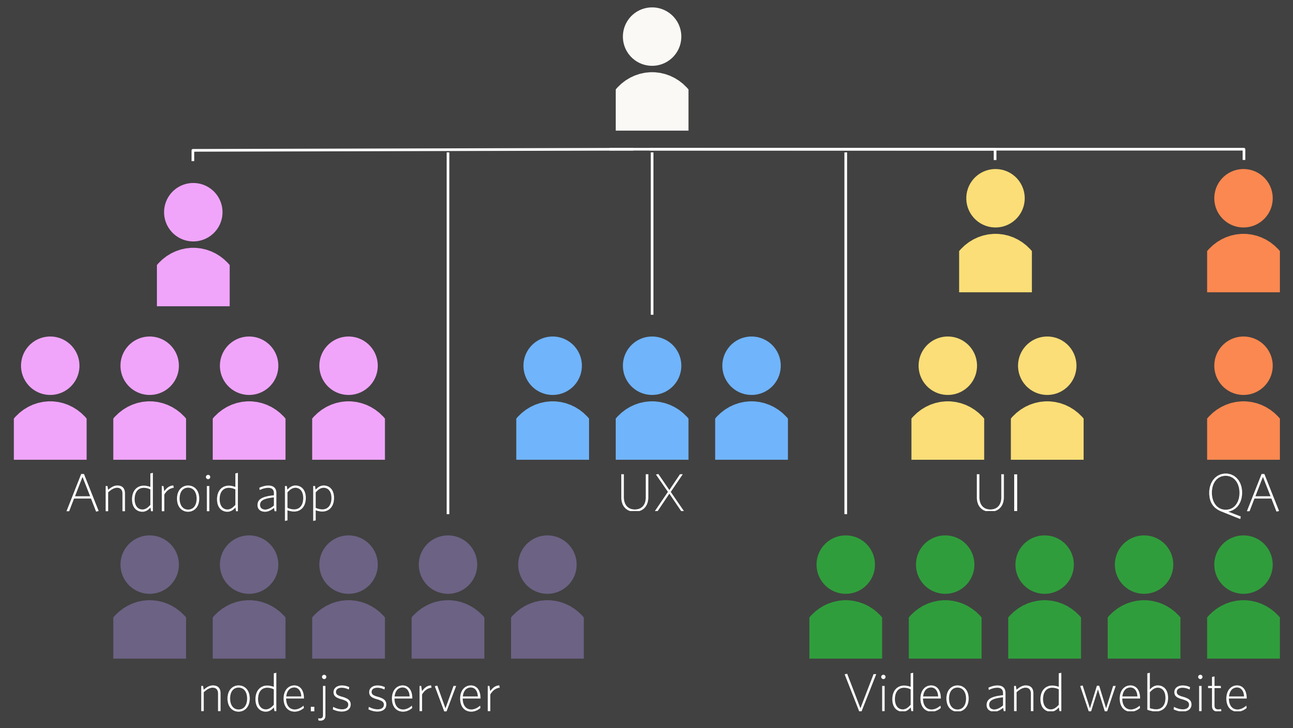
In reality, peer production struggles with tasks that traditional contract-based firms achieve (e.g., marketing, keeping release schedules, integrated contributions). So, hybridized models often support the community.

Example: plugging a USB drive into a Ubuntu machine

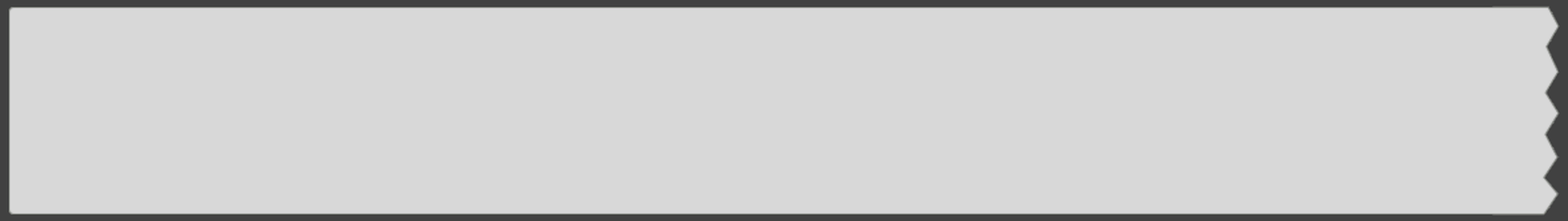
# Flash Organizations

[Valentine et al., CHI '17]

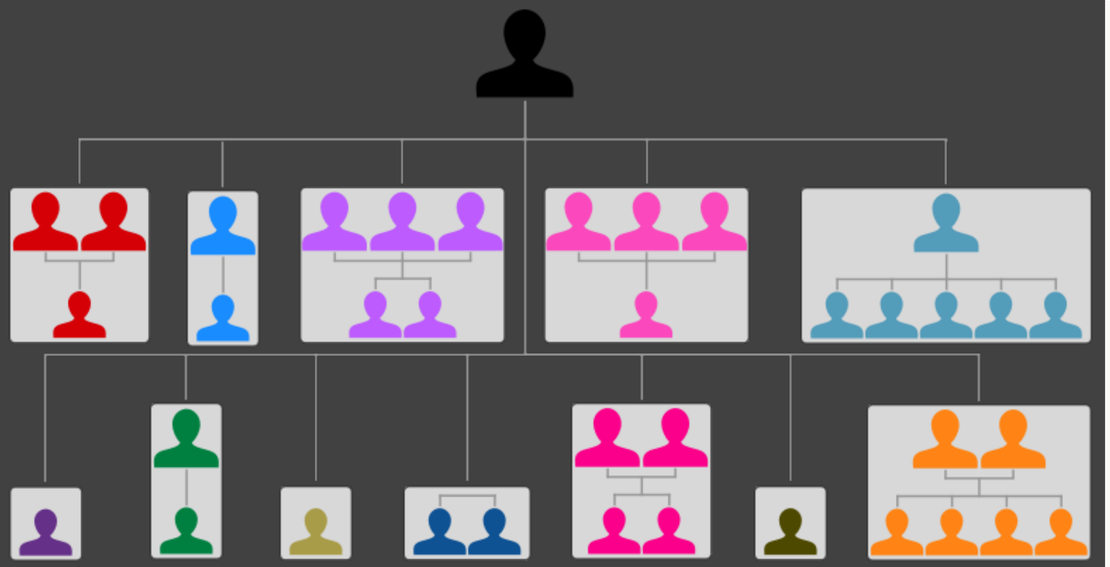
One approach to coordinated adaptation: structuring crowds as computationally-powered organizations, not algorithms



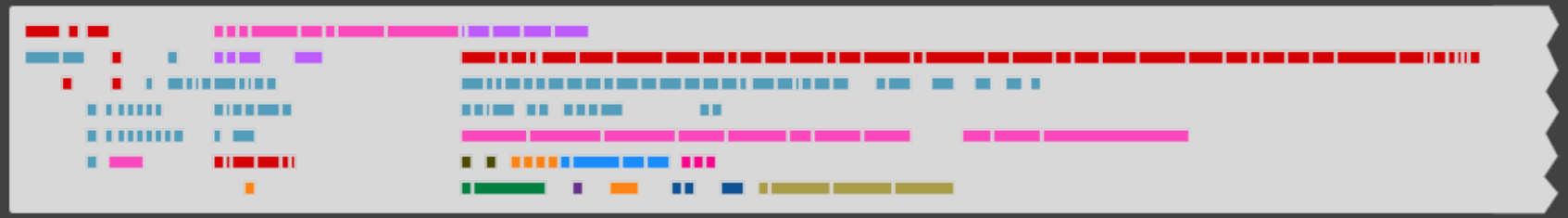
# Example flash organization

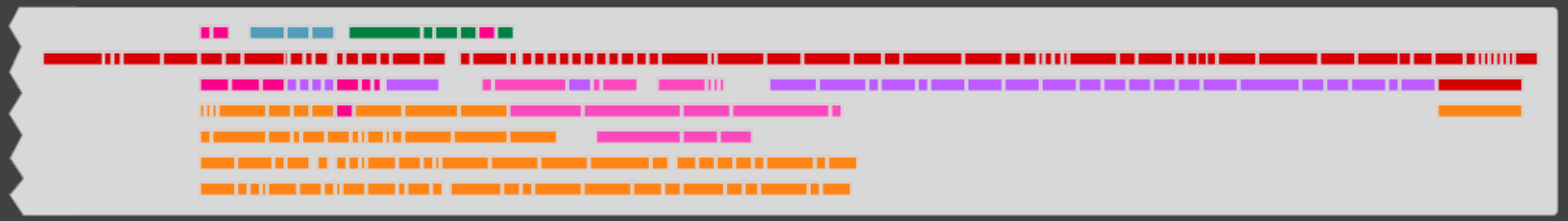


# Example flash organization

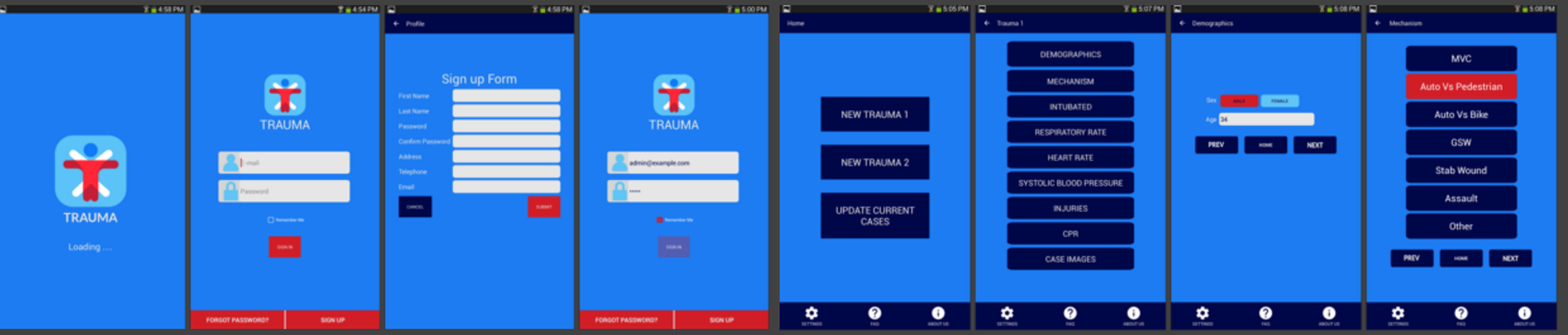


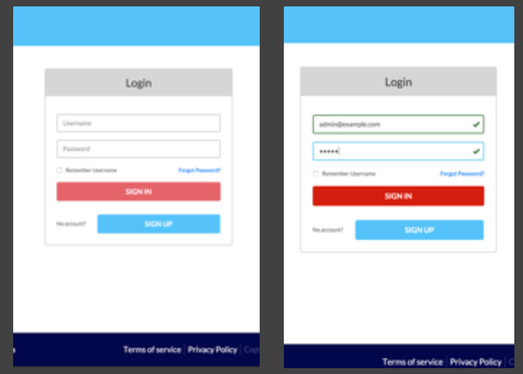
- Android Development
- User Interface Design
- Front End Development
- Back End Development
- User Testing
- HIPAA
- Video
- Market Research
- Marketing Website
- Pitch Deck and User Documentation
- Application Name
- Quality Assurance

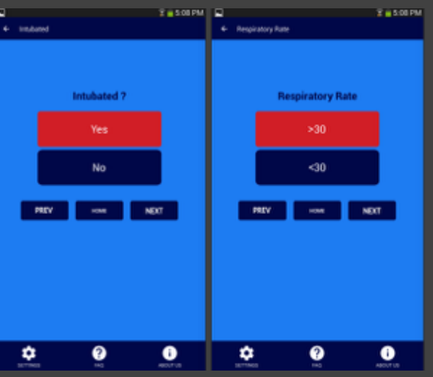




# Example flash organization

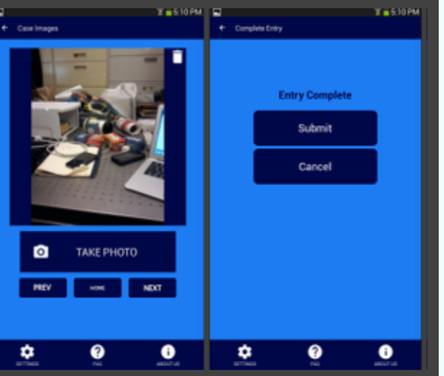




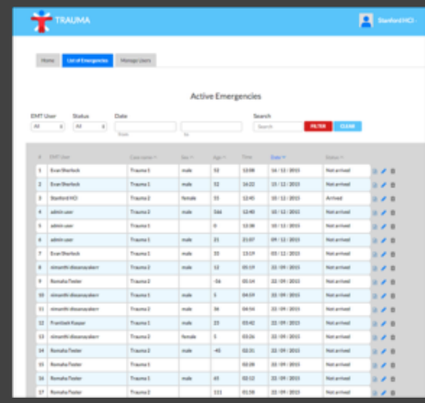








EMS TRAUMA REPORT





# A Class in Two Acts

## Act I: We Got This!

Creating bustling spaces rather than ghost towns
Designing norms and culture
Bootstrapping and prototyping
Growth and breadth
Designing for strong and weak ties
Group collaboration
Wisdom of the crowd
Crowdsourcing and peer production

## Act II: We Don't Got This.

Antisocial computing: mobs and trolls
Unintended consequences
Collective governance
Free speech, ethics, and content moderation
AIs in social environments
Future of work

# Summary

Shifting from simple wisdom-of-the-crowd tasks requires much more than just a scaling up of ambition: it requires designing for interdependence.

Peer production — the term encompassing shared open work (e.g., Wikipedia, open source) is one powerful method for volunteer coordination. Workflows and algorithms offer another approach. Both have their issues.

Aiming higher means we will need to solve issues of convergence and coordinated adapatation.

# Social Computing

CS 278 | Stanford University | Michael Bernstein

Creative Commons images thanks to Kamau Akabueze, Eric Parker, Chris Goldberg, Dick Vos, Wikimedia, MaxPixel.net, Mescon, and Andrew Taylor.

Slide content shareable under a Creative Commons Attribution-NonCommercial 4.0 International License.