

# CS 8803-MDS

# Human-in-the-loop Data

# Analytics

---

Lecture 15

10/12/22

# Logistics

Two papers (10/26 and 11/09) in Part II were updated

For the better!

# Favorite papers

#1 Blink-DB (6 votes)

#2 AQP++ (5 votes)

#3 Auto-suggest (4 votes)

Runner ups: M4, foraging (3 votes)



# Least favorite papers

#1 Microsoft experience (9 votes)

#2 Auto-suggest (3 votes)

Writing matters!



# Today's class

How to make progress in research

Vectoring

Velocity

Part II topics

Hypothesis testing

# What problem are we solving?

“But how do we start?”

“I’m feeling so lost.”

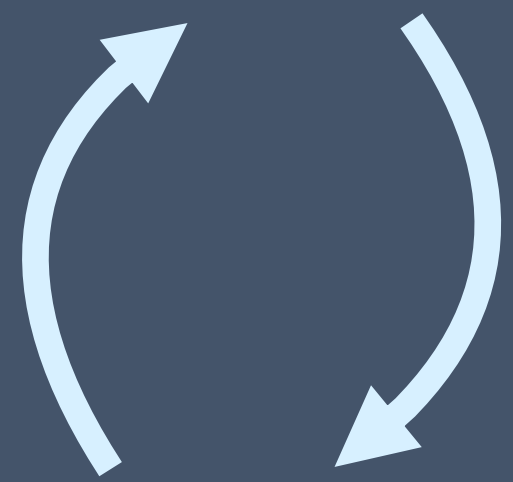
“I thought of an important reason  
that this won’t work.”

“It’s not working yet. I’m not sure that  
we’re making progress.”

# Bernstein theory of faculty success

To be a Stanford-tier faculty member, you need to master two skills that operate in a tight loop with one another.

**Vectoring:** identifying the biggest dimension of risk in your project right now



**Velocity:** rapid reduction of risk in the chosen dimension

# What Is Vectoring?



# What research is not

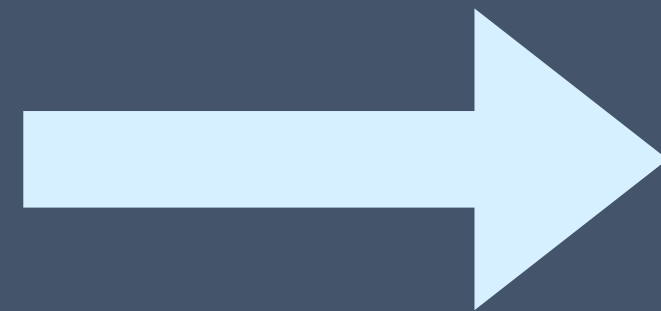
1. Figure out what to do.
2. Do it.
3. Publish.

# What research is

Research is an iterative process of exploration, not a linear path from idea to result [Gowers 2000]

# Problematic points of view

“OK, we have a good idea.  
Let’s build it / model it /  
prove it / get training data.”



“I spent some time thinking  
about this and hacking on it,  
and it’s not going to work:  
it has a fatal flaw.”



**Treating your research  
goal as a project  
specification and  
executing it**

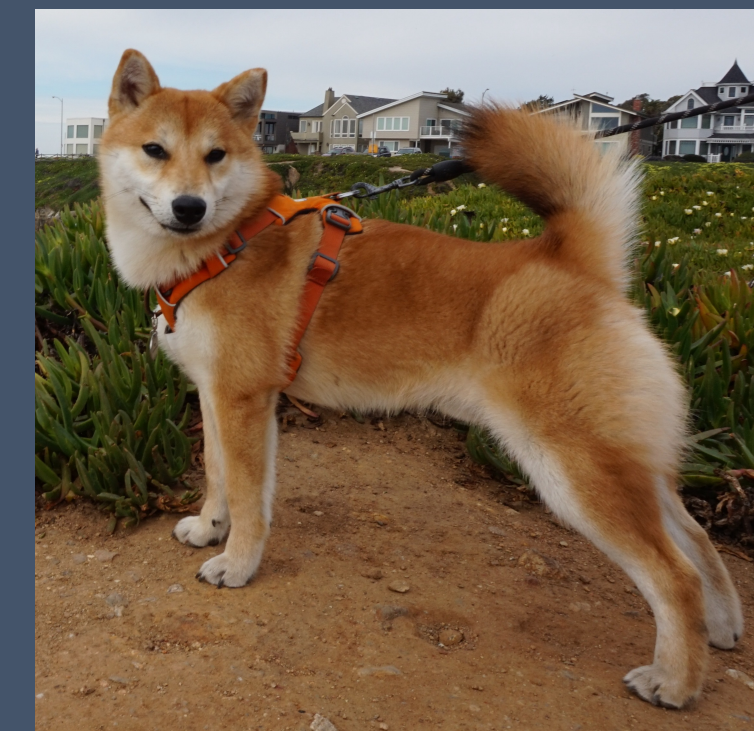
# Idea as project spec

Taking a concept and trying to realize it in parallel across all decisions, assumptions, and goals



Concept

work work work work work work

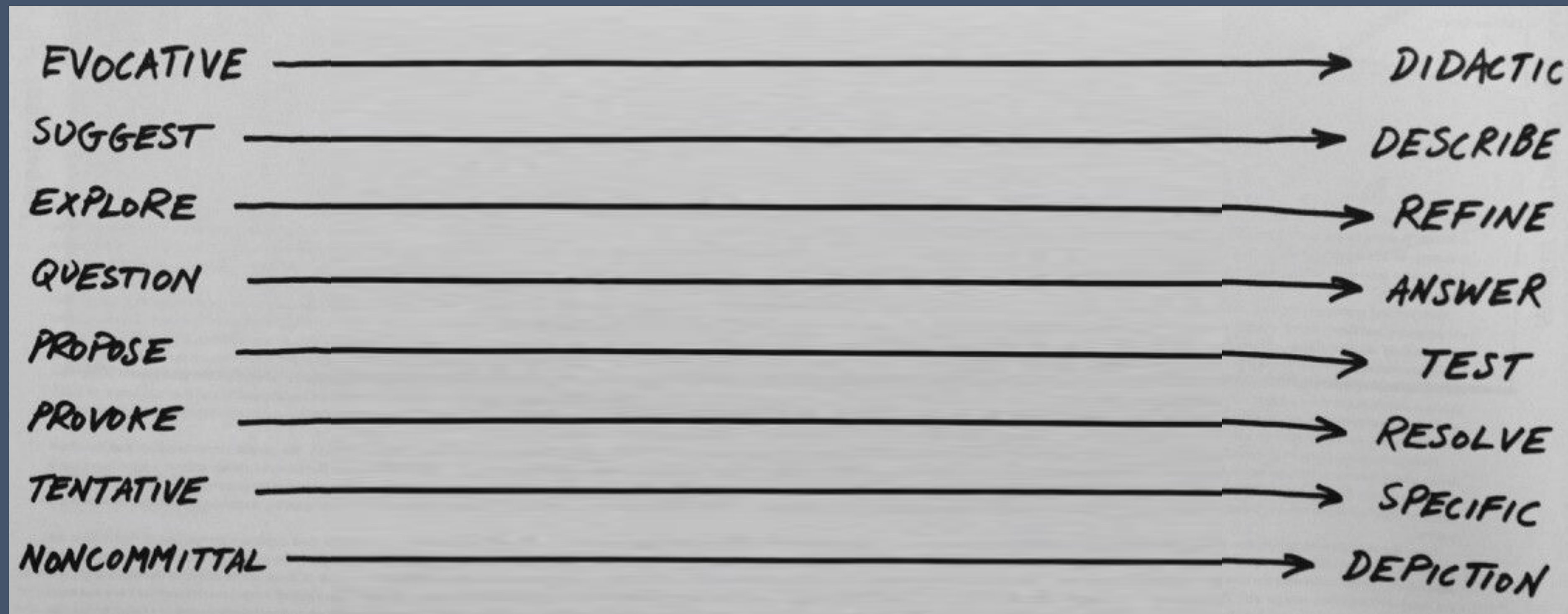


Result

# Idea as project spec

What you should have done

What you did



[Buxton 2007]

This is **all other points**  
of a research project

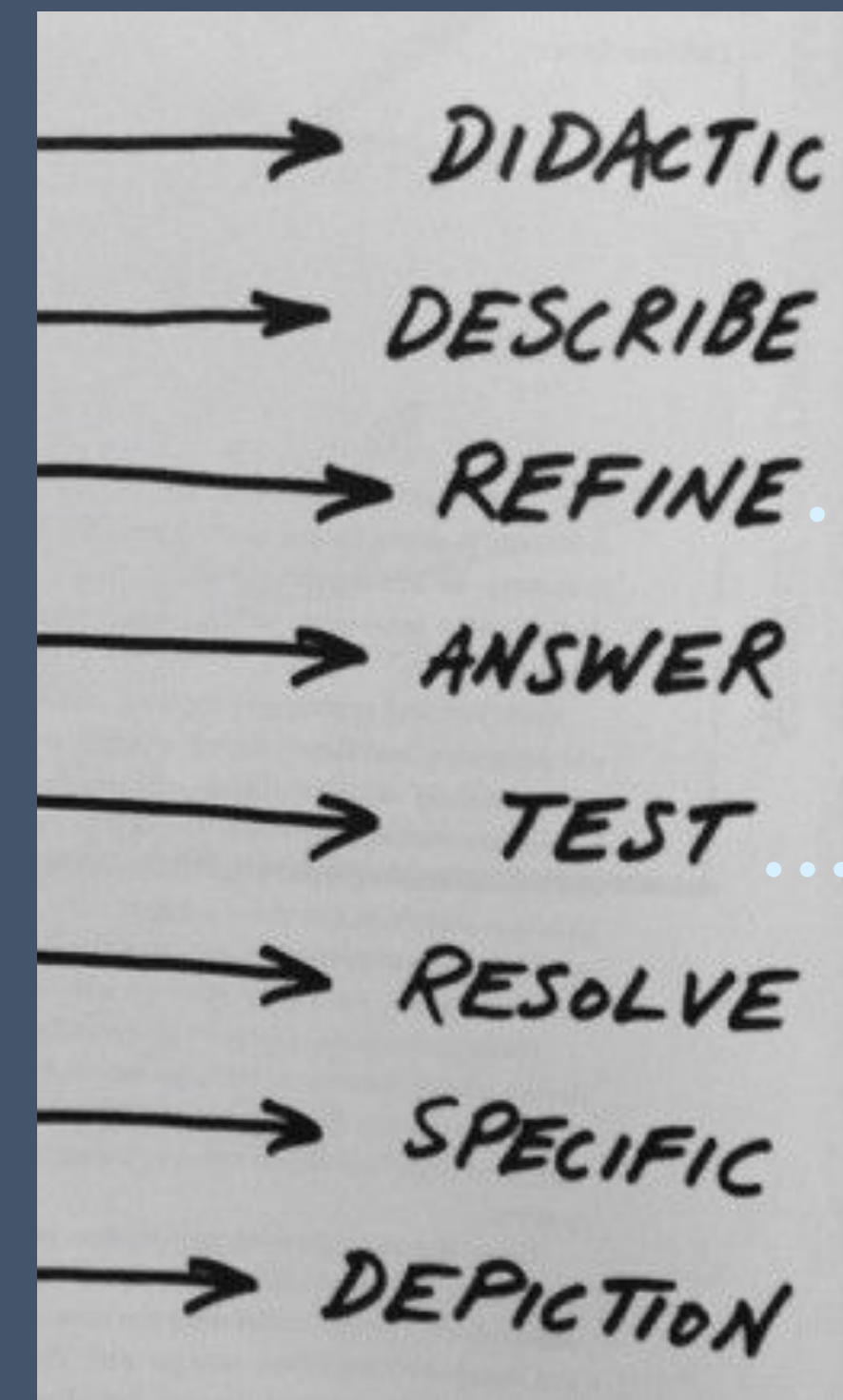
This is the **endpoint**  
of a research project



# Problematic points of view

“OK, we have a good idea.  
Let’s build it / model it /  
prove it / get training data.”

“I spent some time thinking  
about this and hacking on it,  
and it’s not going to work:  
it has a fatal flaw.”

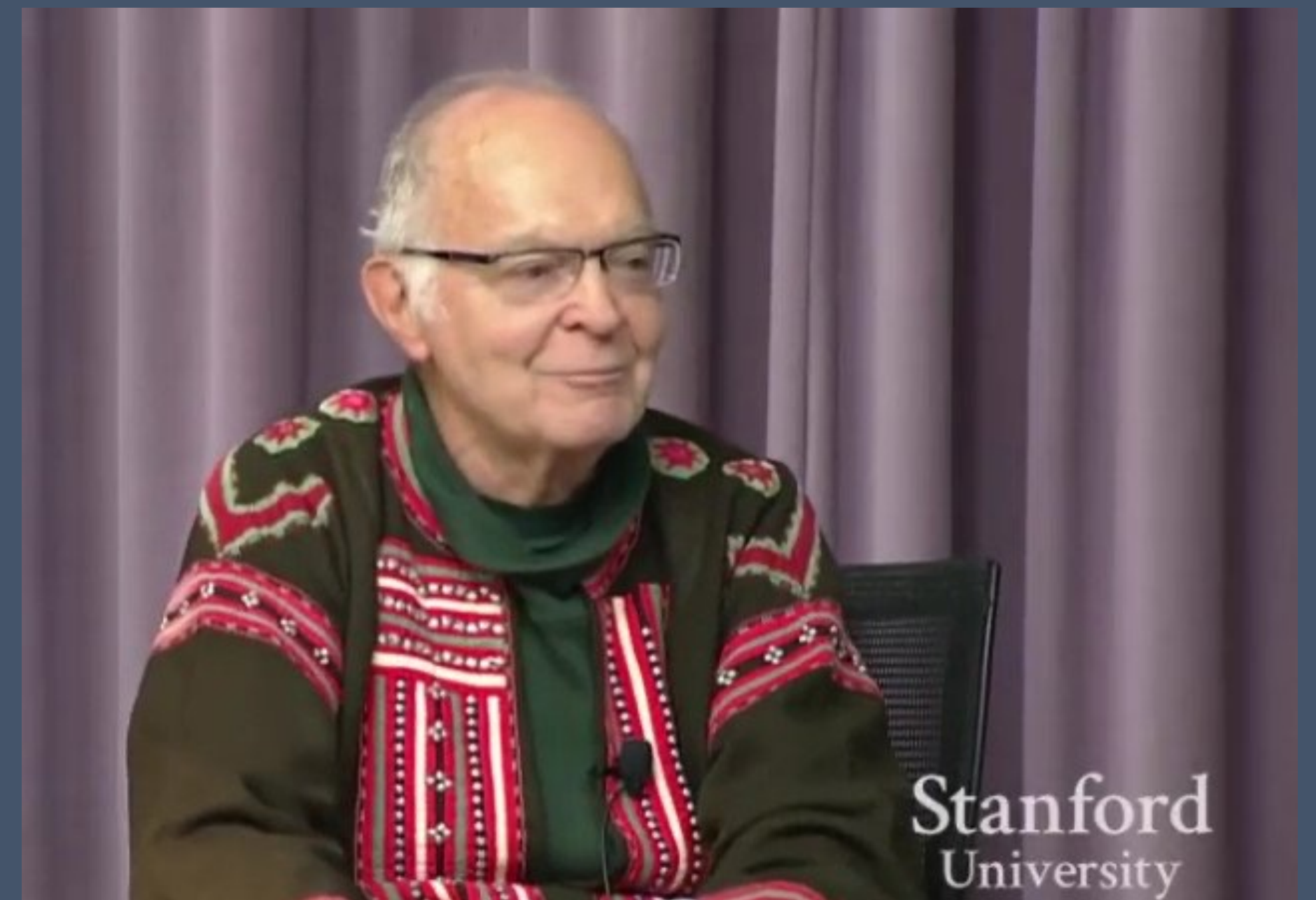


.....before knowing  
what to refine!

.....before identifying  
if that test or flaw is  
the right one to  
focus on!

Premature optimization is the  
root of all evil.

- Donald Knuth



# Pick a vector

It may feel like we get stuck unable to solve the problem because we haven't figured out everything else about it. There are too many open questions, and too many possible directions. The more dimensions there are, the harder gradient descent becomes.

Instead of trying to do everything at once (project spec point of view), pick one dimension of uncertainty — one vector — and focus on reducing its risk and uncertainty.







# Example vectors

**Prototyping:** will this technique work at all? To answer this, we implement a basic version of the technique and mock in the data and other test harness elements.

**Engineering:** will this technique work with a realistic workload? To answer this, we need to engineer a test harness.

**Proving:** does the limit exist that I suspect does? To answer this, we start by writing a proof for a simpler case.

# Implications

The vectors under consideration will each imply building different parts of your system.

Rather than building them all at once, when you might have to change things later, vectoring instead implies that you start by reducing uncertainty in the most important dimension first — your “inner loop” — and then building out from there.

# Vectoring algorithm

## 1. Generate questions

Untested hunches, risky decisions,  
high-level directions

## 2. Rank your questions

Which is most critical?

## 3. Pick one and answer it rapidly

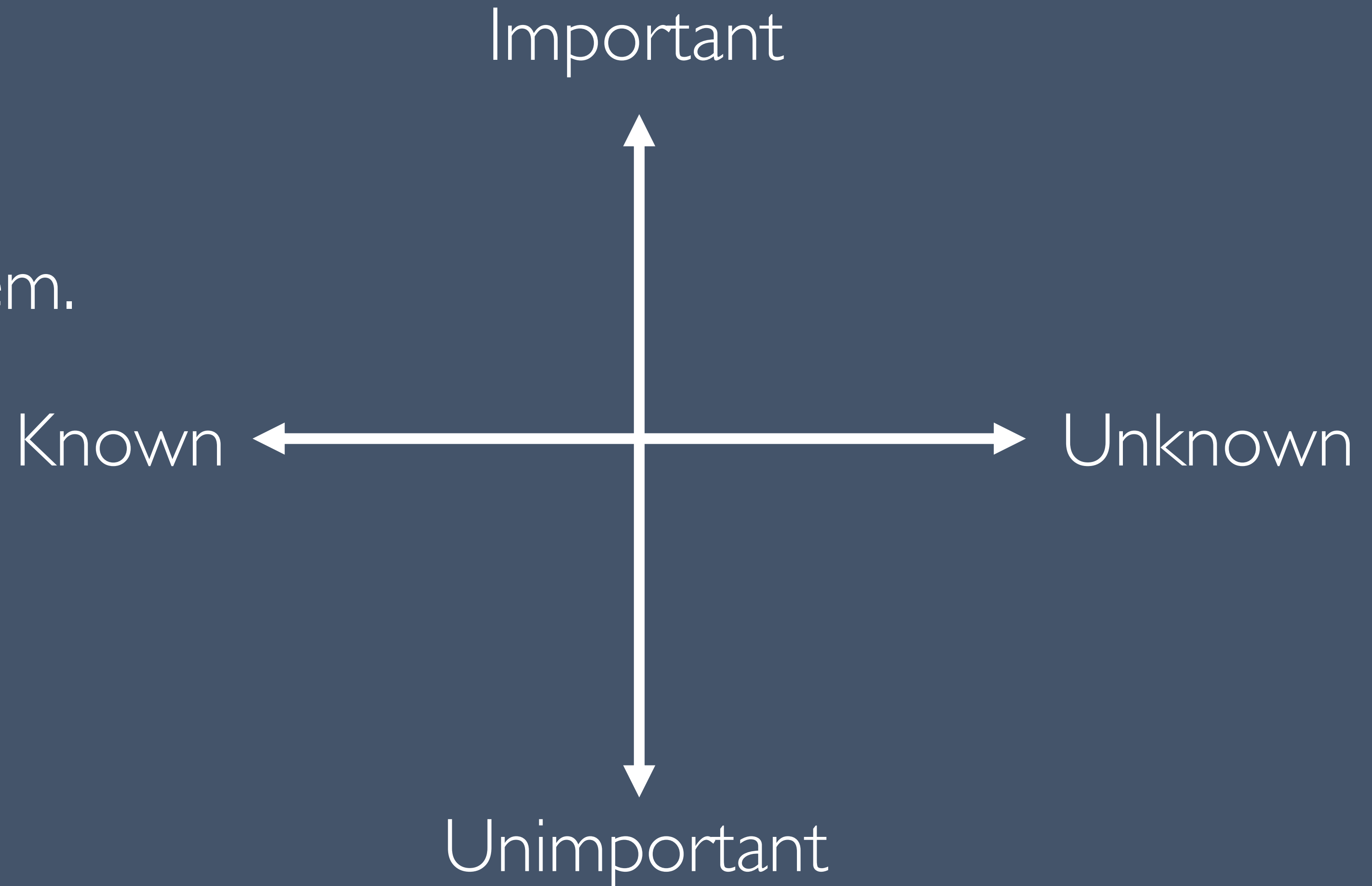
Answer only the most critical question  
(This is where velocity comes into play)



# Assumption mapping

Assumption mapping is a strategy for articulating questions and ranking them.

Try assumption mapping your project [5min]



Let's Try It



# Trolling

While everyone thinks that trolling online is due to a small number of antisocial sociopaths, we had a hunch that “normal” people were responsible for much trolling behavior when triggered.

What’s our first step?

We have: dataset of 16M CNN comments (w/ troll flags)

Adapted from Stanford CS197

## Anyone Can Become a Troll: Causes of Trolling Behavior in Online Discussions

Justin Cheng<sup>1</sup>, Michael Bernstein<sup>1</sup>, Cristian Danescu-Niculescu-Mizil<sup>2</sup>, Jure Leskovec<sup>1</sup>

<sup>1</sup>Stanford University, <sup>2</sup>Cornell University  
{jcccf, msb, jure}@cs.stanford.edu, cristian@cs.cornell.edu

### ABSTRACT

In online communities, antisocial behavior such as trolling disrupts constructive discussion. While prior work suggests that trolling behavior is confined to a vocal and antisocial minority, we demonstrate that ordinary people can engage in such behavior as well. We propose two primary trigger mechanisms: the individual’s mood, and the surrounding context of a discussion (e.g., exposure to prior trolling behavior). Through an experiment simulating an online discussion, we find that both negative mood and seeing troll posts by others significantly increases the probability of a user trolling, and together double this probability. To support and extend these results, we study how these same mechanisms play out in the wild via a data-driven, longitudinal analysis of a large online news discussion community. This analysis reveals temporal mood effects, and explores long range patterns of repeated exposure to trolling. A predictive model of trolling behavior shows that mood and discussion context together can explain trolling behavior better than an individual’s history of trolling. These results combine to suggest that ordinary people can, under the right circumstances, behave like trolls.

### ACM Classification Keywords

H.2.8 Database Management: Database Applications—*Data Mining*; J.4 Computer Applications: Social and Behavioral Sciences

### Author Keywords

Trolling; antisocial behavior; online communities

### INTRODUCTION

As online discussions become increasingly part of our daily interactions [24], antisocial behavior such as trolling [37, 43], harassment, and bullying [82] is a growing concern. Not only does antisocial behavior result in significant emotional distress [1, 58, 70], but it can also lead to offline harassment and threats of violence [90]. Further, such behavior comprises a substantial fraction of user activity on many web sites [18, 24, 30] – 40% of internet users were victims of online harassment [27]; on CNN.com, over one in five comments are removed by moderators for violating community guidelines. What causes this prevalence of antisocial behavior online?

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

In this paper, we focus on the causes of *trolling behavior* in discussion communities, defined in the literature as behavior that falls outside acceptable bounds defined by those communities [9, 22, 37]. Prior work argues that trolls are born and not made: those engaging in trolling behavior have unique personality traits [11] and motivations [4, 38, 80]. However, other research suggests that people can be influenced by their environment to act aggressively [20, 41]. As such, is trolling caused by particularly antisocial individuals or by ordinary people? Is trolling behavior innate, or is it situational? Likewise, what are the conditions that affect a person’s likelihood of engaging in such behavior? And if people can be influenced to troll, can trolling spread from person to person in a community? By understanding what causes trolling and how it spreads in communities, we can design more robust social systems that can guard against such undesirable behavior.

This paper reports a field experiment and observational analysis of trolling behavior in a popular news discussion community. The former allows us to tease apart the causal mechanisms that affect a user’s likelihood of engaging in such behavior. The latter lets us replicate and explore finer grained aspects of these mechanisms as they occur in the wild. Specifically, we focus on two possible causes of trolling behavior: a user’s mood, and the surrounding discussion context (e.g., seeing others’ troll posts before posting).

**Online experiment.** We studied the effects of participants’ prior mood and the context of a discussion on their likelihood to leave troll-like comments. Negative mood increased the probability of a user subsequently trolling in an online news comment section, as did the presence of prior troll posts written by other users. These factors combined to double participants’ baseline rates of engaging in trolling behavior.

**Large-scale data analysis.** We augment these results with an analysis of over 16 million posts on *CNN.com*, a large online news site where users can discuss published news articles. One out of four posts flagged for abuse are authored by users with no prior record of such posts, suggesting that many undesirable posts can be attributed to ordinary users. Supporting our experimental findings, we show that a user’s propensity to troll rises and falls in parallel with known population-level mood shifts throughout the day [32], and exhibits cross-discussion persistence and temporal decay patterns, suggesting that negative mood from bad events linger [41, 45]. Our data analysis also recovers the effect of exposure to prior troll posts in the discussion, and further reveals how the strength of this effect depends on the volume and ordering of these



# Trolling

## Possible vectors:

Do people really troll when angry?

Can we train a classifier to predict when someone would troll, and compare weights of personal history vs. other posts and title?

Does the same person troll more on certain (angry) topics than on other (boring) ones?

Adapted from Stanford CS197

## Anyone Can Become a Troll: Causes of Trolling Behavior in Online Discussions

Justin Cheng<sup>1</sup>, Michael Bernstein<sup>1</sup>, Cristian Danescu-Niculescu-Mizil<sup>2</sup>, Jure Leskovec<sup>1</sup>

<sup>1</sup>Stanford University, <sup>2</sup>Cornell University  
{jcccf, msb, jure}@cs.stanford.edu, cristian@cs.cornell.edu

### ABSTRACT

In online communities, antisocial behavior such as trolling disrupts constructive discussion. While prior work suggests that trolling behavior is confined to a vocal and antisocial minority, we demonstrate that ordinary people can engage in such behavior as well. We propose two primary trigger mechanisms: the individual's mood, and the surrounding context of a discussion (e.g., exposure to prior trolling behavior). Through an experiment simulating an online discussion, we find that both negative mood and seeing troll posts by others significantly increases the probability of a user trolling, and together double this probability. To support and extend these results, we study how these same mechanisms play out in the wild via a data-driven, longitudinal analysis of a large online news discussion community. This analysis reveals temporal mood effects, and explores long range patterns of repeated exposure to trolling. A predictive model of trolling behavior shows that mood and discussion context together can explain trolling behavior better than an individual's history of trolling. These results combine to suggest that ordinary people can, under the right circumstances, behave like trolls.

### ACM Classification Keywords

H.2.8 Database Management: Database Applications—*Data Mining*; J.4 Computer Applications: Social and Behavioral Sciences

### Author Keywords

Trolling; antisocial behavior; online communities

### INTRODUCTION

As online discussions become increasingly part of our daily interactions [24], antisocial behavior such as trolling [37, 43], harassment, and bullying [82] is a growing concern. Not only does antisocial behavior result in significant emotional distress [1, 58, 70], but it can also lead to offline harassment and threats of violence [90]. Further, such behavior comprises a substantial fraction of user activity on many web sites [18, 24, 30] – 40% of internet users were victims of online harassment [27]; on CNN.com, over one in five comments are removed by moderators for violating community guidelines. What causes this prevalence of antisocial behavior online?

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

In this paper, we focus on the causes of *trolling behavior* in discussion communities, defined in the literature as behavior that falls outside acceptable bounds defined by those communities [9, 22, 37]. Prior work argues that trolls are born and not made: those engaging in trolling behavior have unique personality traits [11] and motivations [4, 38, 80]. However, other research suggests that people can be influenced by their environment to act aggressively [20, 41]. As such, is trolling caused by particularly antisocial individuals or by ordinary people? Is trolling behavior innate, or is it situational? Likewise, what are the conditions that affect a person's likelihood of engaging in such behavior? And if people can be influenced to troll, can trolling spread from person to person in a community? By understanding what causes trolling and how it spreads in communities, we can design more robust social systems that can guard against such undesirable behavior.

This paper reports a field experiment and observational analysis of trolling behavior in a popular news discussion community. The former allows us to tease apart the causal mechanisms that affect a user's likelihood of engaging in such behavior. The latter lets us replicate and explore finer grained aspects of these mechanisms as they occur in the wild. Specifically, we focus on two possible causes of trolling behavior: a user's mood, and the surrounding discussion context (e.g., seeing others' troll posts before posting).

**Online experiment.** We studied the effects of participants' prior mood and the context of a discussion on their likelihood to leave troll-like comments. Negative mood increased the probability of a user subsequently trolling in an online news comment section, as did the presence of prior troll posts written by other users. These factors combined to double participants' baseline rates of engaging in trolling behavior.

**Large-scale data analysis.** We augment these results with an analysis of over 16 million posts on *CNN.com*, a large online news site where users can discuss published news articles. One out of four posts flagged for abuse are authored by users with no prior record of such posts, suggesting that many undesirable posts can be attributed to ordinary users. Supporting our experimental findings, we show that a user's propensity to troll rises and falls in parallel with known population-level mood shifts throughout the day [32], and exhibits cross-discussion persistence and temporal decay patterns, suggesting that negative mood from bad events linger [41, 45]. Our data analysis also recovers the effect of exposure to prior troll posts in the discussion, and further reveals how the strength of this effect depends on the volume and ordering of these



# Viz

While most dashboards presents raw time series as they arrive, we had a hunch that some level of smoothing help reduce noise and highlight large-scale trend and deviation.

What's our first step?

Adapted from Stanford CS197

## ASAP: Prioritizing Attention via Time Series Smoothing

Kexin Rong, Peter Bailis

Stanford InfoLab

{krong, pbailis}@cs.stanford.edu

### ABSTRACT

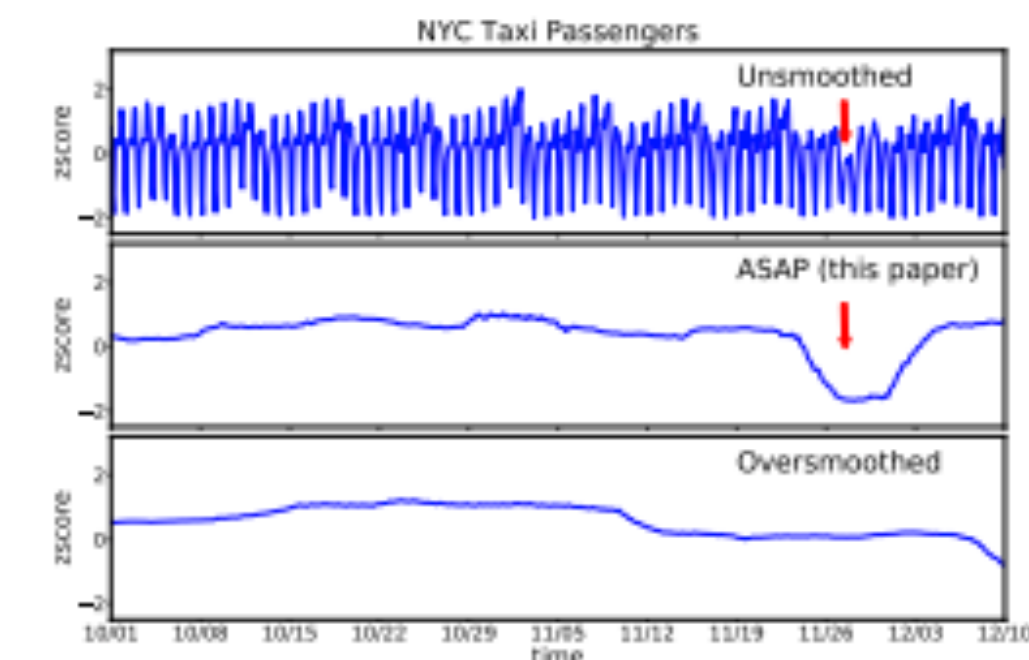
Time series visualization of streaming telemetry (i.e., charting of key metrics such as server load over time) is increasingly prevalent in modern data platforms and applications. However, many existing systems simply plot the raw data streams as they arrive, often obscuring large-scale trends due to small-scale noise. We propose an alternative: to better prioritize end users' attention, smooth time series visualizations as much as possible to remove noise, while retaining large-scale structure to highlight significant deviations. We develop a new analytics operator called ASAP that automatically smooths streaming time series by adaptively optimizing the trade-off between noise reduction (i.e., variance) and trend retention (i.e., kurtosis). We introduce metrics to quantitatively assess the quality of smoothed plots and provide an efficient search strategy for optimizing these metrics that combines techniques from stream processing, user interface design, and signal processing via autocorrelation-based pruning, pixel-aware preaggregation, and on-demand refresh. We demonstrate that ASAP can improve users' accuracy in identifying long-term deviations in time series by up to 38.4% while reducing response times by up to 44.3%. Moreover, ASAP delivers these results several orders of magnitude faster than alternative search strategies.

### 1. INTRODUCTION

Data volumes continue to rise, fueled in large part by an increasing number of automated sources, including sensors, processes, and devices. For example, each of LinkedIn, Twitter, and Facebook reports that their production infrastructure generates over 12M events per second [16, 51, 68]. As a result, the past several years have seen an explosion in the development of platforms for managing, storing, and querying large-scale data streams of time-stamped data—i.e., time series—from on-premises databases including InfluxDB [6], Ganglia [3], Graphite [5], OpenTSDB [9], Prometheus [10], and Facebook Gorilla [51], to cloud services including DataDog [2], New Relic [8], AWS CloudWatch [1], Google Stackdriver [4], and Microsoft Azure Monitor [7]. These time series engines provide application authors, site operators, and “DevOps” engineers a means of performing monitoring, health checks, alerting, and analysis of unusual events such as failures [19, 32].

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>. For any use beyond those covered by this license, obtain permission by emailing [info@vldb.org](mailto:info@vldb.org).

Proceedings of the VLDB Endowment, Vol. 10, No. 11  
Copyright 2017 VLDB Endowment 2150-8097/17/07.



**Figure 1:** Normalized number of NYC taxi passengers over 10 weeks.<sup>1</sup> From top to bottom, the three plots show the hourly average (unsmoothed), the weekly average (smoothed) and the monthly average (oversmoothed) of the same time series. The arrows point to the week of Thanksgiving (11/27), when the number of passengers dips. This phenomenon is most prominent in the smoothed plot produced by ASAP, the subject of this paper.

While these engines have automated and optimized common tasks in the storage and processing of time series, effective visualization of time series remains a challenge. Specifically, in conversations with engineers using time series data and databases in cloud services, social networking, industrial manufacturing, electrical utilities, and mobile applications, we learned that many production time series visualizations (i.e., “dashboards”) simply display raw data streams as they arrive. Engineers reported this display of raw data can be a poor match for production scenarios involving data exploration and debugging. That is, as data arrives in increasing volumes, even small-scale fluctuations in data values can obscure overall trends and behavior. For example, an electrical utility employs two staff to perform 24-hour monitoring of generators. It is critical that these staff quickly identify any systematic shifts of generator metrics in their monitoring dashboards, even those that are “sub-threshold” with respect to a critical alarm. Unfortunately, such sub-threshold events are easily obscured by short-term fluctuations in the visualization.

The resulting challenge in time series visualization at scale is presenting the *appropriate* plot that prioritizes users’ attention towards significant deviations. To illustrate this challenge using public data, consider the time series depicted in Figure 1. The top plot shows raw

<sup>1</sup> Here and later in this paper, we depict z-scores [40] instead of raw values. This choice of visualization provides a means of normalizing the visual field across plots while still highlighting large-scale trends.



# Viz

Possible vectors:

Can we find datasets in which a phenome/trend is only visible after smoothing?

Can we find datasets that are best remain unsmoothed?

## ASAP: Prioritizing Attention via Time Series Smoothing

Kexin Rong, Peter Bailis

Stanford InfoLab

{krong, pbailis}@cs.stanford.edu

### ABSTRACT

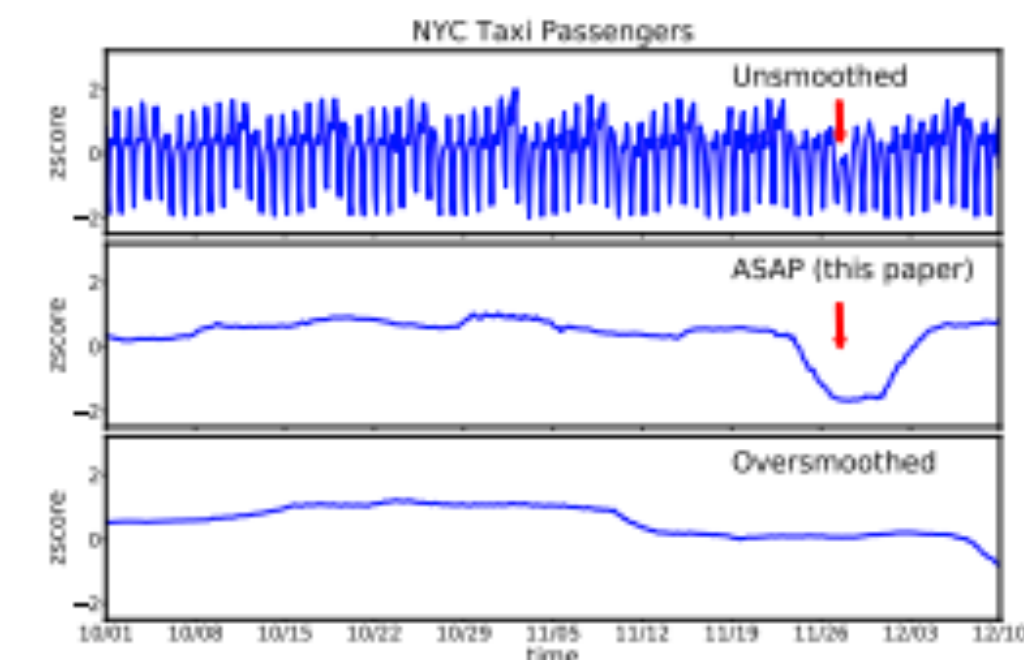
Time series visualization of streaming telemetry (i.e., charting of key metrics such as server load over time) is increasingly prevalent in modern data platforms and applications. However, many existing systems simply plot the raw data streams as they arrive, often obscuring large-scale trends due to small-scale noise. We propose an alternative: to better prioritize end users' attention, smooth time series visualizations as much as possible to remove noise, while retaining large-scale structure to highlight significant deviations. We develop a new analytics operator called ASAP that automatically smooths streaming time series by adaptively optimizing the trade-off between noise reduction (i.e., variance) and trend retention (i.e., kurtosis). We introduce metrics to quantitatively assess the quality of smoothed plots and provide an efficient search strategy for optimizing these metrics that combines techniques from stream processing, user interface design, and signal processing via autocorrelation-based pruning, pixel-aware preaggregation, and on-demand refresh. We demonstrate that ASAP can improve users' accuracy in identifying long-term deviations in time series by up to 38.4% while reducing response times by up to 44.3%. Moreover, ASAP delivers these results several orders of magnitude faster than alternative search strategies.

### 1. INTRODUCTION

Data volumes continue to rise, fueled in large part by an increasing number of automated sources, including sensors, processes, and devices. For example, each of LinkedIn, Twitter, and Facebook reports that their production infrastructure generates over 12M events per second [16, 51, 68]. As a result, the past several years have seen an explosion in the development of platforms for managing, storing, and querying large-scale data streams of time-stamped data—i.e., time series—from on-premises databases including InfluxDB [6], Ganglia [3], Graphite [5], OpenTSDB [9], Prometheus [10], and Facebook Gorilla [51], to cloud services including DataDog [2], New Relic [8], AWS CloudWatch [1], Google Stackdriver [4], and Microsoft Azure Monitor [7]. These time series engines provide application authors, site operators, and “DevOps” engineers a means of performing monitoring, health checks, alerting, and analysis of unusual events such as failures [19, 32].

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>. For any use beyond those covered by this license, obtain permission by emailing [info@vldb.org](mailto:info@vldb.org).

Proceedings of the VLDB Endowment, Vol. 10, No. 11  
Copyright 2017 VLDB Endowment 2150-8097/17/07.



**Figure 1:** Normalized number of NYC taxi passengers over 10 weeks.<sup>1</sup> From top to bottom, the three plots show the hourly average (unsmoothed), the weekly average (smoothed) and the monthly average (oversmoothed) of the same time series. The arrows point to the week of Thanksgiving (11/27), when the number of passengers dips. This phenomenon is most prominent in the smoothed plot produced by ASAP, the subject of this paper.

While these engines have automated and optimized common tasks in the storage and processing of time series, effective visualization of time series remains a challenge. Specifically, in conversations with engineers using time series data and databases in cloud services, social networking, industrial manufacturing, electrical utilities, and mobile applications, we learned that many production time series visualizations (i.e., “dashboards”) simply display raw data streams as they arrive. Engineers reported this display of raw data can be a poor match for production scenarios involving data exploration and debugging. That is, as data arrives in increasing volumes, even small-scale fluctuations in data values can obscure overall trends and behavior. For example, an electrical utility employs two staff to perform 24-hour monitoring of generators. It is critical that these staff quickly identify any systematic shifts of generator metrics in their monitoring dashboards, even those that are “sub-threshold” with respect to a critical alarm. Unfortunately, such sub-threshold events are easily obscured by short-term fluctuations in the visualization.

The resulting challenge in time series visualization at scale is presenting the *appropriate* plot that prioritizes users' attention towards significant deviations. To illustrate this challenge using public data, consider the time series depicted in Figure 1. The top plot shows raw

<sup>1</sup> Here and later in this paper, we depict z-scores [40] instead of raw values. This choice of visualization provides a means of normalizing the visual field across plots while still highlighting large-scale trends.

Why is vectoring so  
important?

“If Ernest Hemingway, James Mitchener, Neil Simon, Frank Lloyd Wright, and Pablo Picasso could not get it right the first time, what makes you think that you will?”

— Paul Heckel



# Iteration >> planning

Ideas rarely land exactly where you expect they will. It's best to test the most critical assumptions quickly, so that you can understand whether your hunch will play out, and what problems are worth spending time solving vs. kludging.

Human creative work is best in a loop of reflection and iteration. Vectoring is a way to make sure you're getting the most iteration cycles.

# Re-vectoring

Often, after vectoring and reducing uncertainty in one dimension, it raises new questions and uncertainties.

In the next round of vectoring, you re-prioritize:

If you get unexpected results and are confused (most of the time!), maybe it means you take a new angle to reduce uncertainty on a vector related to the prior one.

If you answer your question to your own satisfaction (not completely, just to your satisfaction), you move on to the next most important vector

# Magnitude of your vector

The result of vectoring should be something achievable in about a week's sprint. If it's not, you've picked too broad a question to answer.

If your vectoring for “Can normal people be responsible for a lot of the trolling online?” is “Can normal people be responsible for a lot of the trolling on CNN.com?”, you're still way too broad.

That's evidence that you've just rescaled your project,  not picked a vector.

**Takeaways, in brief**

1) The temptation is to try and solve the problem that's set in front of you. Don't.



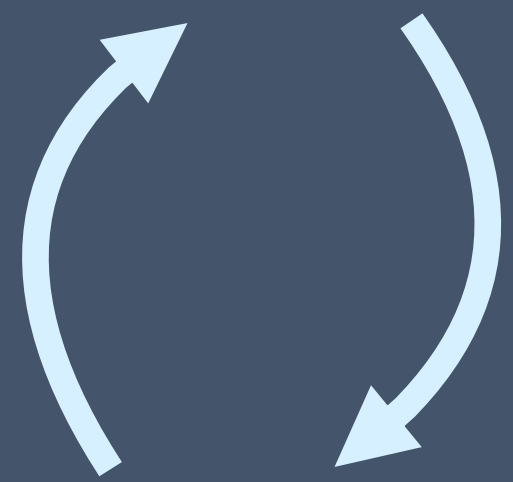
2) Vectoring is a process of identifying the dimension of highest impact+uncertainty, and prioritizing that dimension while scaffolding the others

3) Successful vectoring enables you to rapidly hone in on the core insight of your research project

# Bernstein theory of faculty success

To be a Stanford-tier faculty member, you need to master two skills that operate in a tight loop with one another.

**Vectoring:** identifying the biggest dimension of risk in your project right now



**Velocity:** rapid reduction of risk in the chosen dimension

# What Is Velocity?

# Problematic point of view

“Research is so much slower than industry.”



“I feel like we’re just not getting anywhere.”



**We’re not making enough progress.**

“I missed another submission deadline.”





# My diagnosis: The Swamp

Nearly every project has a swamp.

**The Swamp:** challenges that get the project stuck for an extended length of time

Model not performing well

Design not having intended effect

Engineering challenges keep cropping up

&etc



Photo by Big Cypress National Preserve

# Swamps make progress a poor measure

Swamps can make a project appear to have no or little progress for an extended period of time.

**However, swamps are when you need to be at your most creative.** You need to try many different ideas, and rapidly, to orienteer your way out of a swamp.

The difference between an amazing and a merely good researcher: **how effectively and rapidly you explore ways to escape the swamp.**



# Enter velocity

Drawn from theory and practice of rapid prototyping

Buxton, Sketching User Experiences

Schön, The Reflective Practitioner

Houde and Hill, What Do Prototypes Prototype?

**“Enlightened trial and error succeeds over the planning of the lone genius.”** - Tom Kelley



# Velocity vs. progress

Progress is an absolute delta of your position from the last time we met. How far have you gotten?

Velocity is a measure of the **distance traveled** in that time.

**If you tried a ton of creative different ideas and they all failed...**

that's low progress  
but high velocity



I will be thrilled.

# Why is velocity a better measure?

Because we have likely learned a ton from the failures along the way.

Because we likely needed to experience those failures to eventually get to a success: you're learning the landscape.

**Because the worst outcome is not failure, but tunneling unproductively.**

That's low progress  
and low velocity



this is when I will  
be disappointed.

How do I achieve  
high velocity?

# Restating our goal, precisely

We have a question to answer this week:

Will our hunch work in a simple case? Is assumption  $X$  valid? Will this revised model overcome the problematic issue? Can we write a proof for the simple case?

We've chosen this week's question that we're trying to answer carefully.

**Velocity is the process of answering that question as rapidly as possible.**

Choosing this question is the process of vectoring.

# Approach: core vs. periphery

Achieving high velocity means sprinting to answer this week's question, while minimizing all other desiderata for now.

This means being clear with yourself on what you can ignore:

**Core:** the goal that needs to be achieved in order to answer the question

**Periphery:** the goals that can be faked, or assumed, or subsetted, or mocked in, so we can focus on the core.

# Core-periphery mindset

The week's goal is an **answer to a question**.

To answer a question, you don't need to address all the issues in the periphery. Just focus on what's in the core.

Make strong assumptions about everything that's in the periphery: use a smaller subset of the data, make simplifying assumptions while working on your proof, ignore other nagging questions for the moment



# Core-periphery mindset

Your approach should be, necessarily, incomplete. Do not create a mockup or a demo. Instead, derive everything from your current question:

- Will this measure correlate with my gut observations?

- Will this engineering approach be satisfactory?

Be rapid. Be ruthless. Strip out or fake everything not required to answer the question.

# Core-periphery mindset

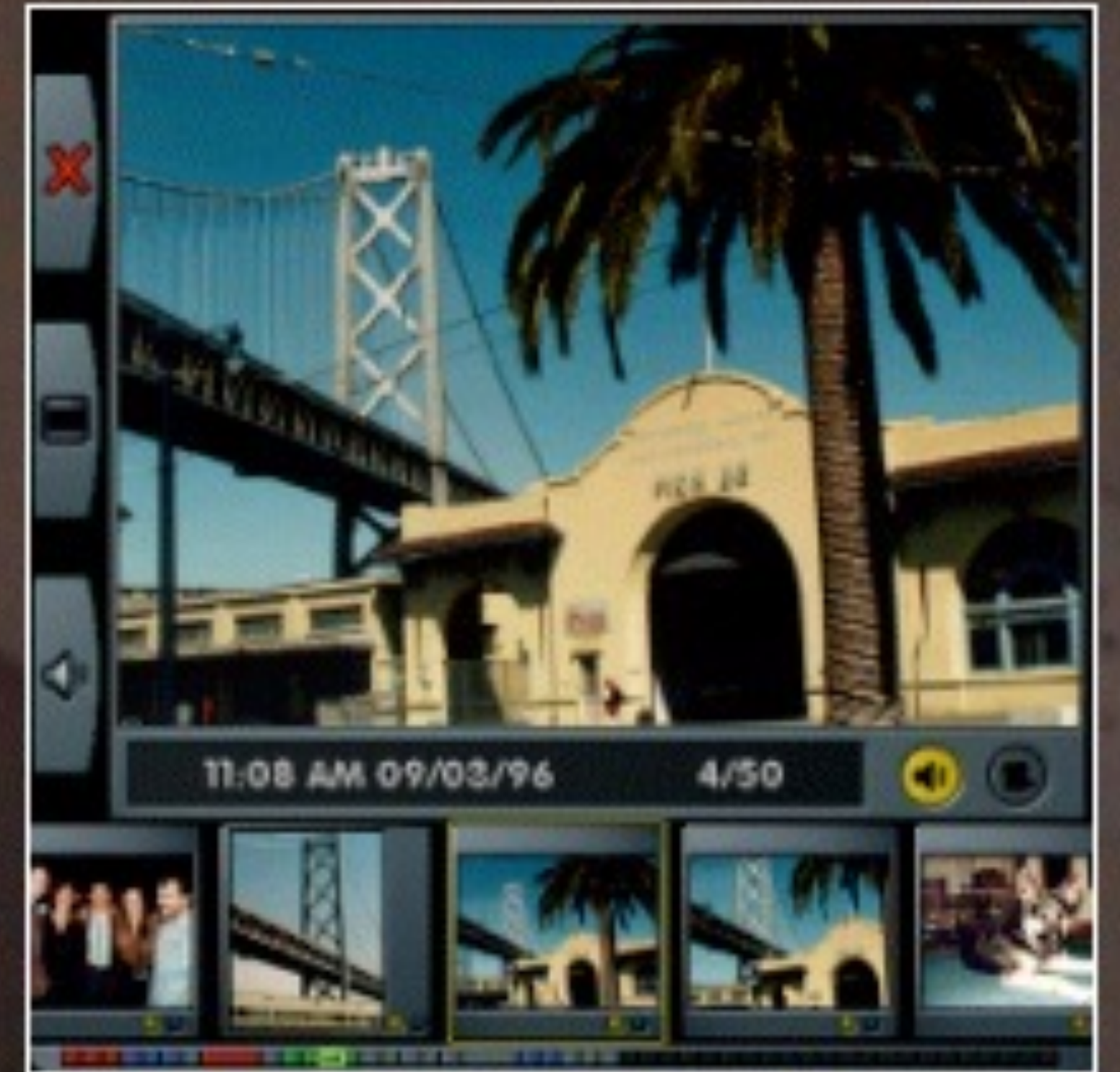
Seriously: I'm dedicating a third slide to this.

Answer questions, don't engineer. This tends to rankle essentially every facet of your undergraduate training.

Too often, people pursue perfection in the first pass: perfect drafts, perfectly engineered software, perfect interaction design.

Remember: the goal is to answer the question, not to build that part of your system permanently (yet).





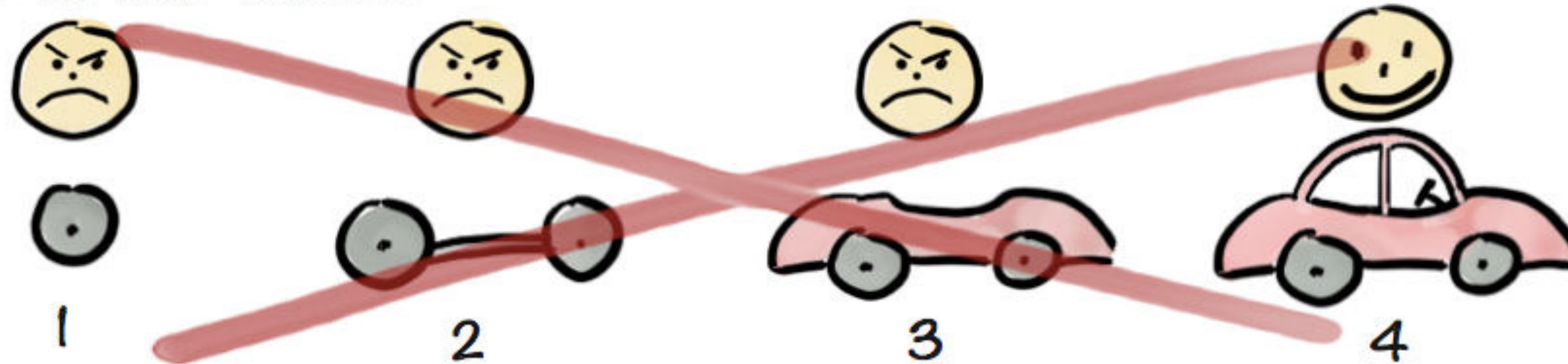
What question  
were they asking?

What did they  
trade off?

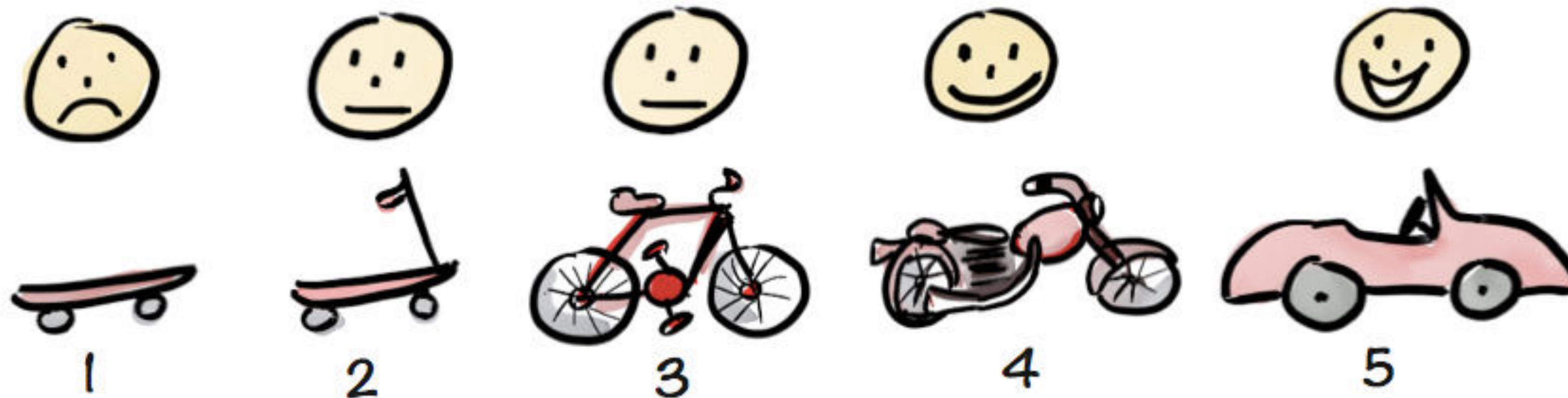


# “Minimal Viable Product”

Not like this....



Like this!



by Henrik Kniberg

# All together now

Each week, we engage in vectoring to identify the biggest unanswered question. This should be the focus of your velocity sprint for the week.

To hit high velocity, be strategic about stripping out all other dependencies, faking what you need to, etc., in order to answer the question.

Be prepared to iterate multiple times within the week!

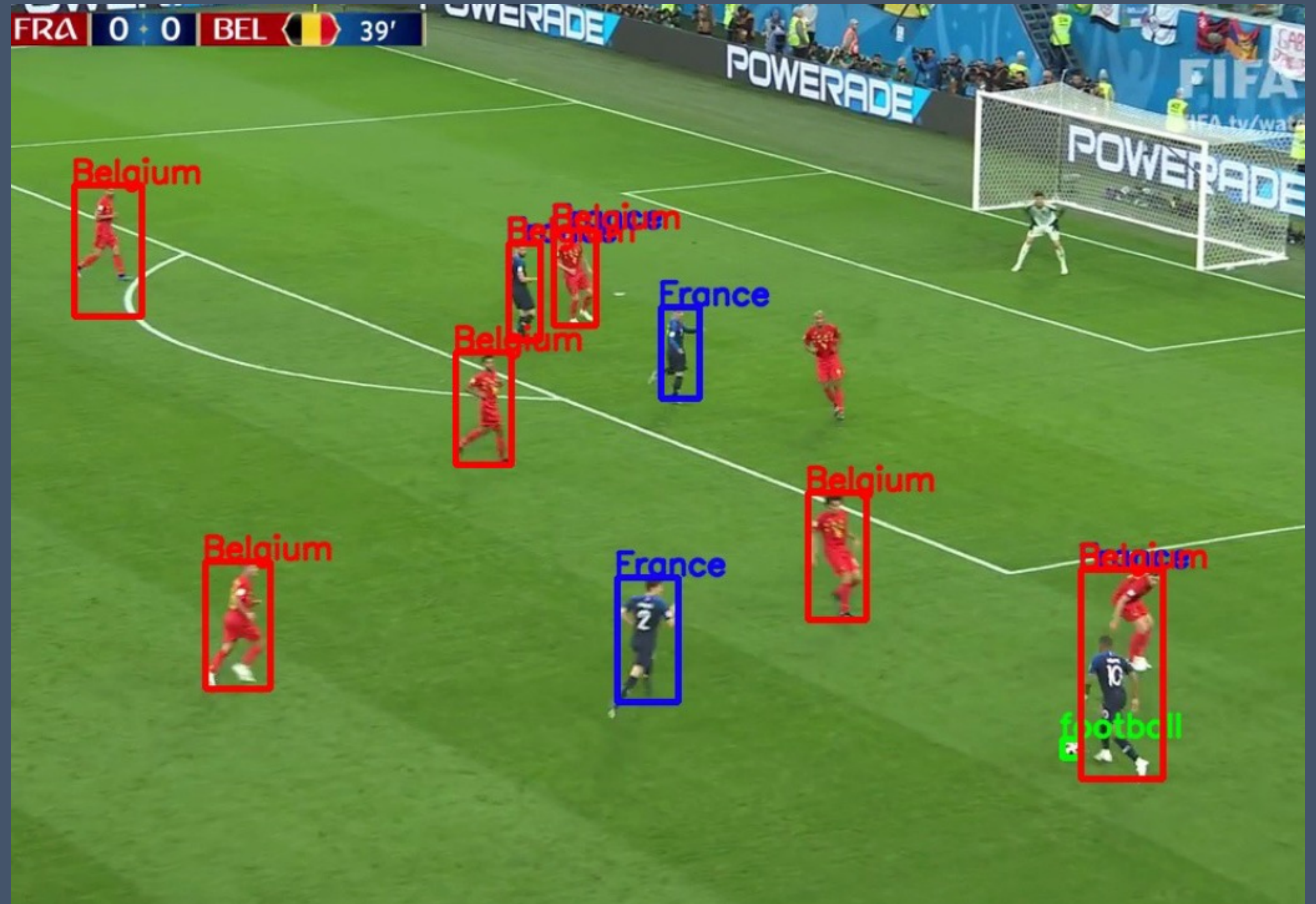
Let's Try It



# Visual query interface

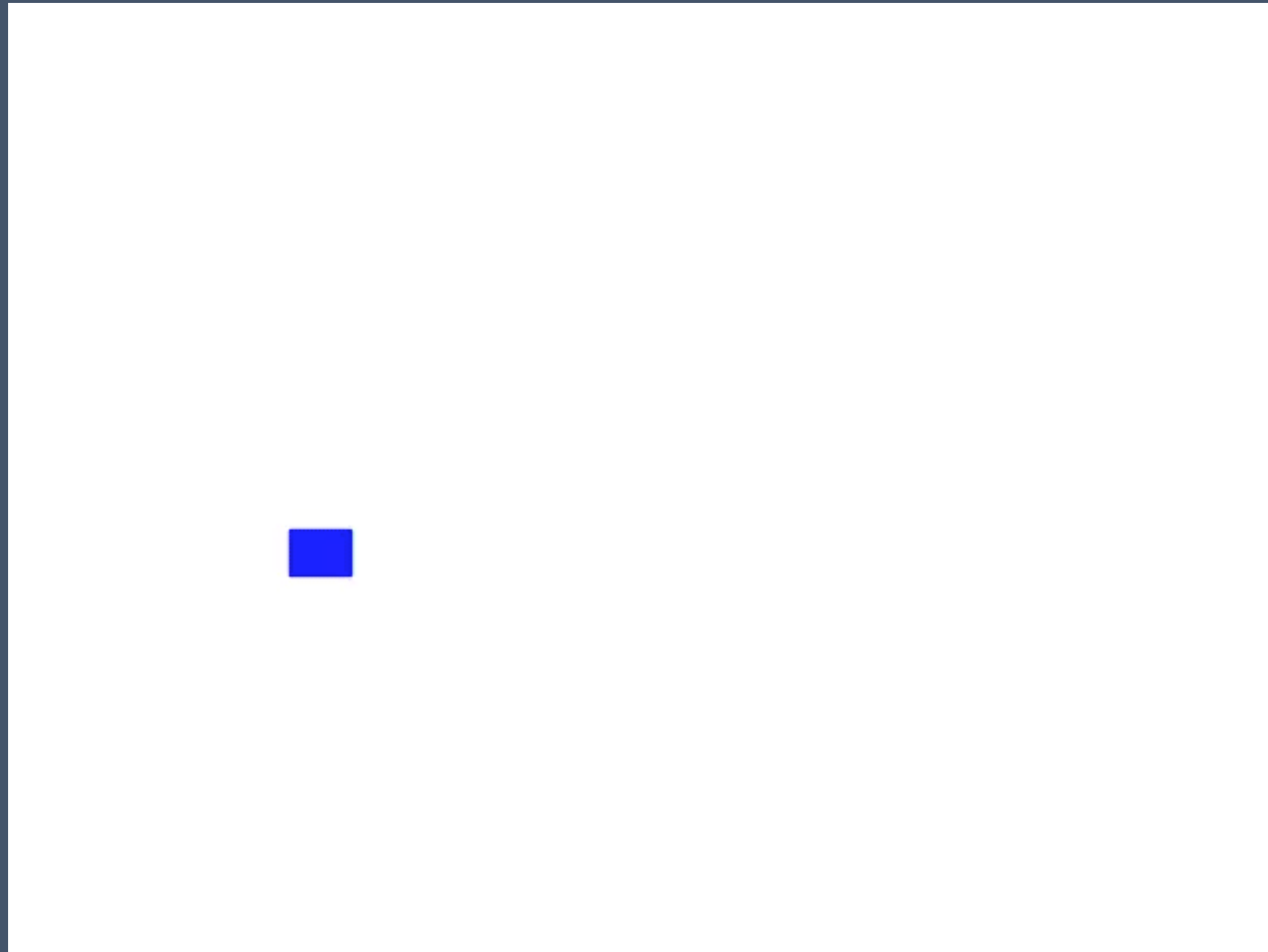
We want to enable users to query videos by drawing out how entities move around in the screen space.

We have object detection labels on the video, but we don't have a query interface yet.

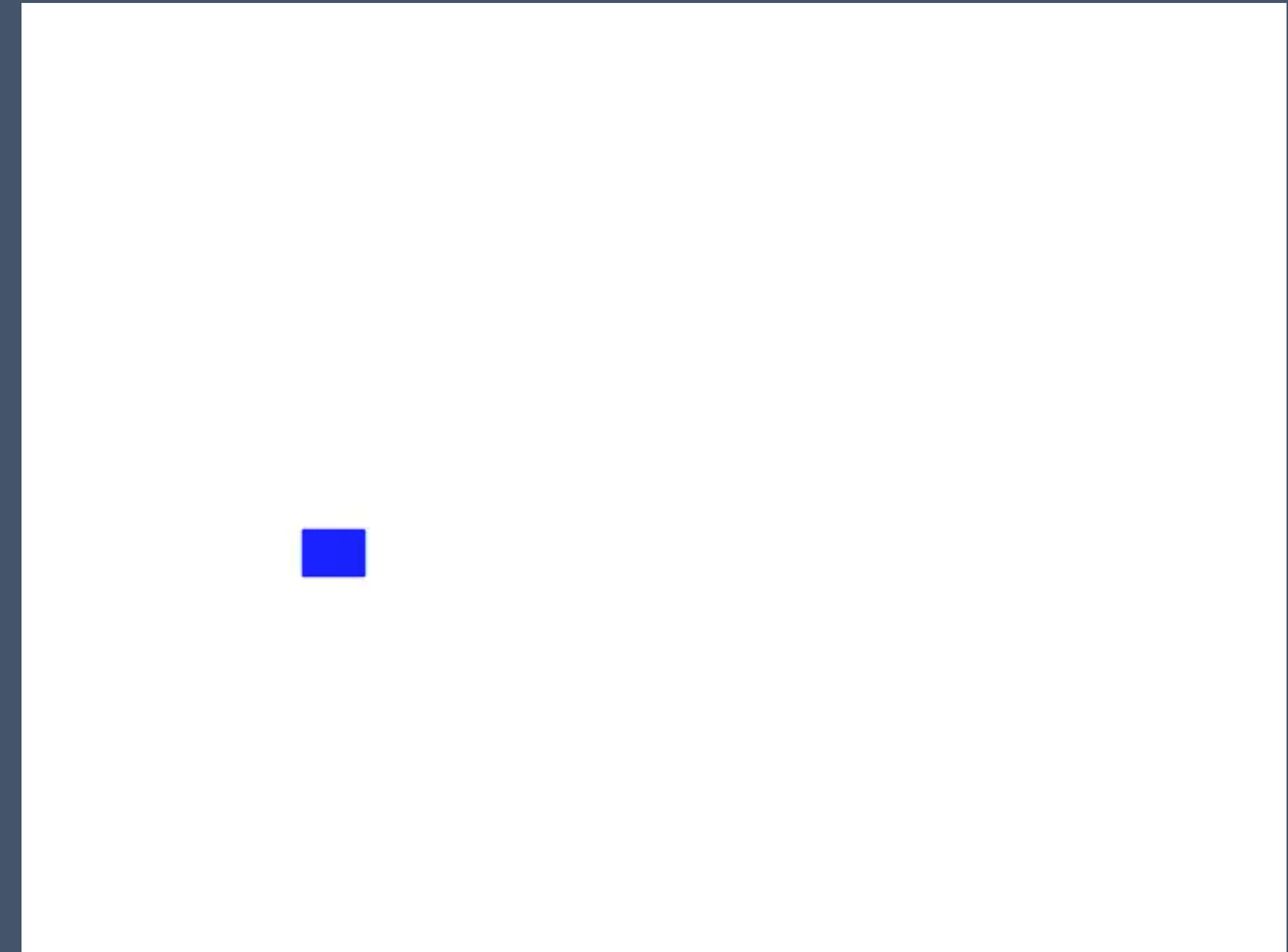


# Visual query interface

We hand crafted (hard coded) several queries.



Two players running in parallel



Car left turn



# Dream Team

This project used multi-armed bandits to identify over several rounds of interaction whether teams should be flat or hierarchical, supportive or critical, etc. But we didn't know: could these multi-armed bandits actually converge fast enough to be useful?

We had a rough implementation of the multi-armed bandits, but it wasn't production ready for interacting with teams.

Adapted from Stanford CS197

**In Search of the Dream Team:  
Temporally Constrained Multi-Armed Bandits for  
Identifying Effective Team Structures**

Sharon Zhou, Melissa Valentine, Michael S. Bernstein  
Stanford University  
sharonz@cs.stanford.edu, mav@stanford.edu, msb@cs.stanford.edu




Figure 1. Each team succeeds under different roles, norms, and interaction patterns; there are no universally ideal team structures. The DreamTeam system exposes teams to a series of different team structures over time to identify effective structures for each team, based on feedback. We introduce multi-armed bandits with temporal constraints to guide this exploration without overwhelming teams in a deluge of simultaneous changes.

**ABSTRACT**  
Team structures—roles, norms, and interaction patterns—define how teams work. HCI researchers have theorized ideal team structures and built systems nudging teams towards them, such as those increasing turn-taking, deliberation, and knowledge distribution. However, organizational behavior research argues against the existence of universally ideal structures. Teams are diverse and excel under different structures: while one team might flourish under hierarchical leadership and a critical culture, another will flounder. In this paper, we present *DreamTeam*: a system that explores a large space of possible team structures to identify effective structures for each team based on observable feedback. To avoid overwhelming teams with too many changes, DreamTeam introduces *multi-armed bandits with temporal constraints*: an algorithm that manages the timing of exploration-exploitation trade-offs across multiple bandits simultaneously. A field experiment demonstrated that DreamTeam teams outperformed self-managing teams by 38%, manager-led teams by 46%, and teams with unconstrained bandits by 41%. This research advances computation as a powerful partner in establishing effective teamwork.

**ACM Classification Keywords**  
H.5.3 Group and Org. Interfaces: Collaborative computing.

**Author Keywords**  
Teams; technical social computing; multi-armed bandits.

**INTRODUCTION**  
Human-computer interaction research has featured a long line of systems that influence teams' roles, norms, and interaction patterns. Roles, norms, and interaction patterns—known collectively as *team structures*—define how a team works together [32]. For many years, HCI researchers have theorized ideal team structures [1, 45] and built systems that nudge teams toward those structures, such as by increasing shared awareness [18, 20], adding channels of communication [65, 64, 70], and convening effective collaborators [38, 50]. The result is a literature that empowers ideal team structures. However, organizational behavior research denies the existence of universally ideal team structures [53, 3, 4, 26]. Structural contingency theory [17] has demonstrated that the best team structures depend on the task, the members, and other factors. This begs the question: when should a team favor one team structure over another? Should the team have centralized or decentralized hierarchy? Should it enforce equal participation from each member? Should members offer each other more encouraging or critical feedback? The wrong decisions can doom a team to dysfunction [32, 53, 3, 4]. Even highly-paid experts—managers—struggle to pick effective team structures [15]. They are hardly to blame, as the set of possibilities is vast [29], with lengthy volumes, dedicated

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.  
CHI 2018, April 21–26, 2018, Montreal, QC, Canada.  
© 2018 Copyright held by the owner(s). Publication rights licensed to ACM.  
ISBN 978-1-4503-5620-6/18/04...\$15.00  
DOI: <https://doi.org/10.1145/3173574.3173682>



# Dream Team

We used a rough simulation! Assuming some roughly accurate numbers in how much each team benefited from each bandit setting, we generated teams and simulated the bandits over a few rounds.

The answer: they converged quickly enough that this might work!

(The next step: wizard of oz the interface, so we could test it “for real” without building integrating software.)

Adapted from Stanford CS197

**In Search of the Dream Team:  
Temporally Constrained Multi-Armed Bandits for  
Identifying Effective Team Structures**

Sharon Zhou, Melissa Valentine, Michael S. Bernstein  
Stanford University  
sharonz@cs.stanford.edu, mav@stanford.edu, msb@cs.stanford.edu




Figure 1. Each team succeeds under different roles, norms, and interaction patterns; there are no universally ideal team structures. The DreamTeam system exposes teams to a series of different team structures over time to identify effective structures for each team, based on feedback. We introduce multi-armed bandits with temporal constraints to guide this exploration without overwhelming teams in a deluge of simultaneous changes.

**ABSTRACT**  
Team structures—roles, norms, and interaction patterns—define how teams work. HCI researchers have theorized ideal team structures and built systems nudging teams towards them, such as those increasing turn-taking, deliberation, and knowledge distribution. However, organizational behavior research argues against the existence of universally ideal structures. Teams are diverse and excel under different structures: while one team might flourish under hierarchical leadership and a critical culture, another will flounder. In this paper, we present *DreamTeam*: a system that explores a large space of possible team structures to identify effective structures for each team based on observable feedback. To avoid overwhelming teams with too many changes, DreamTeam introduces *multi-armed bandits with temporal constraints*: an algorithm that manages the timing of exploration–exploitation trade-offs across multiple bandits simultaneously. A field experiment demonstrated that DreamTeam teams outperformed self-managing teams by 38%, manager-led teams by 46%, and teams with unconstrained bandits by 41%. This research advances computation as a powerful partner in establishing effective teamwork.

**ACM Classification Keywords**  
H.5.3 Group and Org. Interfaces: Collaborative computing.

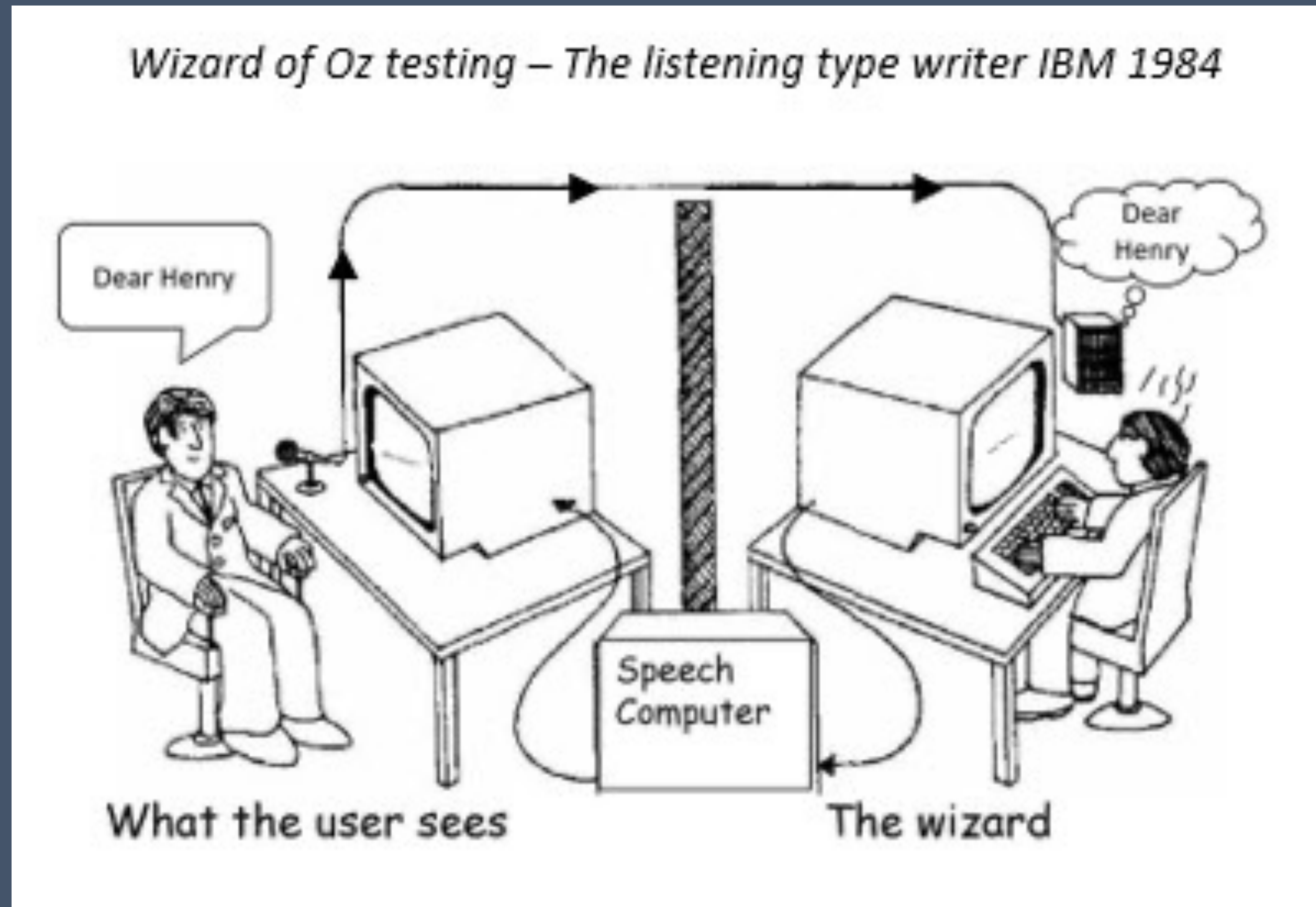
**Author Keywords**  
Teams; technical social computing; multi-armed bandits.

**INTRODUCTION**  
Human-computer interaction research has featured a long line of systems that influence teams’ roles, norms, and interaction patterns. Roles, norms, and interaction patterns—known collectively as *team structures*—define how a team works together [32]. For many years, HCI researchers have theorized ideal team structures [1, 45] and built systems that nudge teams toward those structures, such as by increasing shared awareness [18, 20], adding channels of communication [65, 64, 70], and convening effective collaborators [38, 50]. The result is a literature that empowers ideal team structures. However, organizational behavior research denies the existence of universally ideal team structures [53, 3, 4, 26]. Structural contingency theory [17] has demonstrated that the best team structures depend on the task, the members, and other factors. This begs the question: when should a team favor one team structure over another? Should the team have centralized or decentralized hierarchy? Should it enforce equal participation from each member? Should members offer each other more encouraging or critical feedback? The wrong decisions can doom a team to dysfunction [32, 53, 3, 4]. Even highly-paid experts—managers—struggle to pick effective team structures [15]. They are hardly to blame, as the set of possibilities is vast [29], with lengthy volumes, dedicated

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.  
CHI 2018, April 21–26, 2018, Montreal, QC, Canada.  
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ISBN 978-1-4503-5620-6/18/04...\$15.00  
DOI: <https://doi.org/10.1145/3173574.3173682>



# Wizard of Oz testing



A method of testing a system that does not yet exist.

# Your turn

Pair up with someone NOT on your project.

3min each person: describe your project's current state, the current question you're trying answer. Brainstorm together how to increase velocity.

Afterwards, we'll share out.

# A reminder: the algorithm

1. Articulate the question you're answering.
2. Decide what's absolutely core to answering that question.
3. Decide what's peripheral.
4. Decide the level of fidelity that is absolutely necessary.
5. Go — but be open to reevaluating your assumptions as you go.
6. Loop with a new question.

# Tips and tricks



# “I’m being low velocity.”

**Velocity = distance / time**

So, if your velocity is low, you have two options:

**1. Cover more distance:** habits that can get you further in the same time (e.g., “try harder”, “be a better engineer”)

You’re typically already maxed out on this.

**2. Decrease the time:** prototype more effectively

WIN. Prototype more narrowly, lower your fidelity expectations (e.g., spit out any draft)

# Checking email or social media?

This signals a lack of focus, and is a pretty certain predictor that you're in a swamp.



It means you're prototyping too broadly: **you're unfocused!**  
**focus your goal.** Or you're requiring too high a level of fidelity:  
**you have unreasonable standards! lower your expectations.**

Develop an internal velocity sensor, and as soon as you recognize this, apply one of the two rules.

# Lowering standards: parallelism

Too often, we suffer from what's known in the literature as **fixation**: being certain in an idea and pursuing it to the exclusion of all else. We cannot separate ego from artifact.

Instead, to answer the question, it's often best to **explore multiple approaches in parallel**.

“While the quantity group was busily churning out piles of work—and learning from their mistakes—the quality group had sat theorizing about perfection, and in the end had little more to show for their efforts than grandiose theories and a pile of dead clay.”

— Bayles and Orland, 2001

Adapted from Stanford CS197

# Corollary I: pivoting

Velocity is why cutting yourself off short and pivoting to a new project can be so dangerous in research.

Typically people pivot after a week in the swamp (the “fatal flaw fallacy”), rather than iterating with high velocity out of the swamp.

I promise that the project you pivot to will have a swamp too. Learn to increase velocity and prototype your way out of the swamp faster, instead of seeking out a swampless project.



# Corollary 2: technical debt

Obviously, at some point you need to make sure you're not too deep in technical debt, design debt, or writing debt.

But luckily, **most people can only run their processors hot for a few hours a day**. Everything I've described takes a lot out of you.

When you're out of creative cycles, spend time maturing other parts of your project that are no longer open questions. Or, sometimes we reach a phase where we pause prototyping and focus on refinement and execution for a bit.

Why is velocity so  
important?

# Great research requires high velocity

Don't let 6-12 month paper deadlines obscure the velocity at which research needs to move in order to succeed.

**If you want to achieve a high impact idea, you need to try a lot of approaches and refine and fail a lot.** You want to do that as quickly as possible.

If you can prototype and learn and fail 5x as quickly as the next person, you will be able to achieve far more risky and impactful research.

Takeaways, in brief



1) The swamp is real, and it slows visible progress.

2) Velocity is a far better measure of yourself than progress, and it's something you actually have control over.

3) Achieve high velocity by being clear what question you're answering, and focusing ruthlessly on the core of that question while stripping out the periphery.

4) If you're low velocity,  
 $\text{velocity} = \text{distance} / \text{time}$ . Either  
increase distance (rarely possible) or  
decrease time (often possible: you're  
too broad or too perfectionist).



# And finally..... progress report

Each week for the next several weeks, your team will perform vectoring, submit a brief summary and slide, and report in section:

This week's vector

This week's plan

This week's result

Next week's vector

Next week's plan

# Next class

No class on Monday! Happy fall break

MacroBase: Prioritizing Attention in Fast Data

Authors: Haotian, Yiheng

Reviewer: Eric

Archaeologist: Qiandong

Researcher: Cuong