September 2024

Ongoing Research on Software Engineering Productivity

DPE Summit 2024

Simon Obstbaum
Yegor Denisov-Blanch
Stanford University

Stanford University

Research Team (not exhaustive)

Simon Obstbaum

- Ex-CTO, Crunchyroll & Ellation
  - Portfolio of video streaming services, 100M+ users
  - Hundreds of engineers across 20+ distributed teams
- Founder & ex-CEO, YOPESO (software dev. house, 250+ engineers) (exited)

Yegor Denisov-Blanch

- Stanford Graduate Researcher since 2022
- Research focus: data-driven decision-making in software engineering
- Digital transformation for F100 company with 6,000+ engineers

Prof. Michal Kosinski

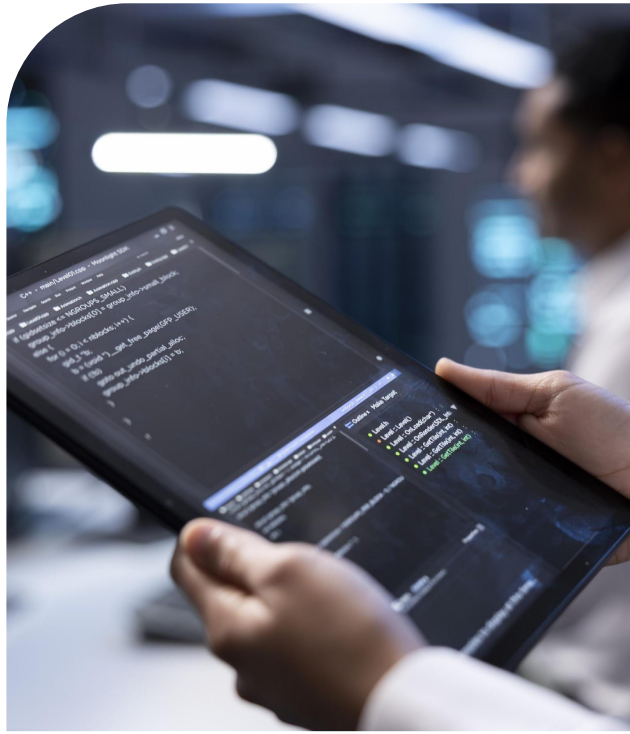
- Stanford Professor (top 1% most cited researchers)
- Research focus: human behavior in a digital environment
- Cambridge Analytica whistleblower
- Stanford Computer Science Postdoc

What we'll cover

1. Motivation Behind Our Research
2. Output vs Outcomes
3. Our Latest Research
4. Case Studies
5. Ongoing Research

What are people using today to measure team productivity?

- Lines of Code
- # of Commits/PRs
- Story Points
- Surveys
- DORA



Existing methods don’t accurately measure productivity

LoC, Commits, and PRs don’t measure productivity

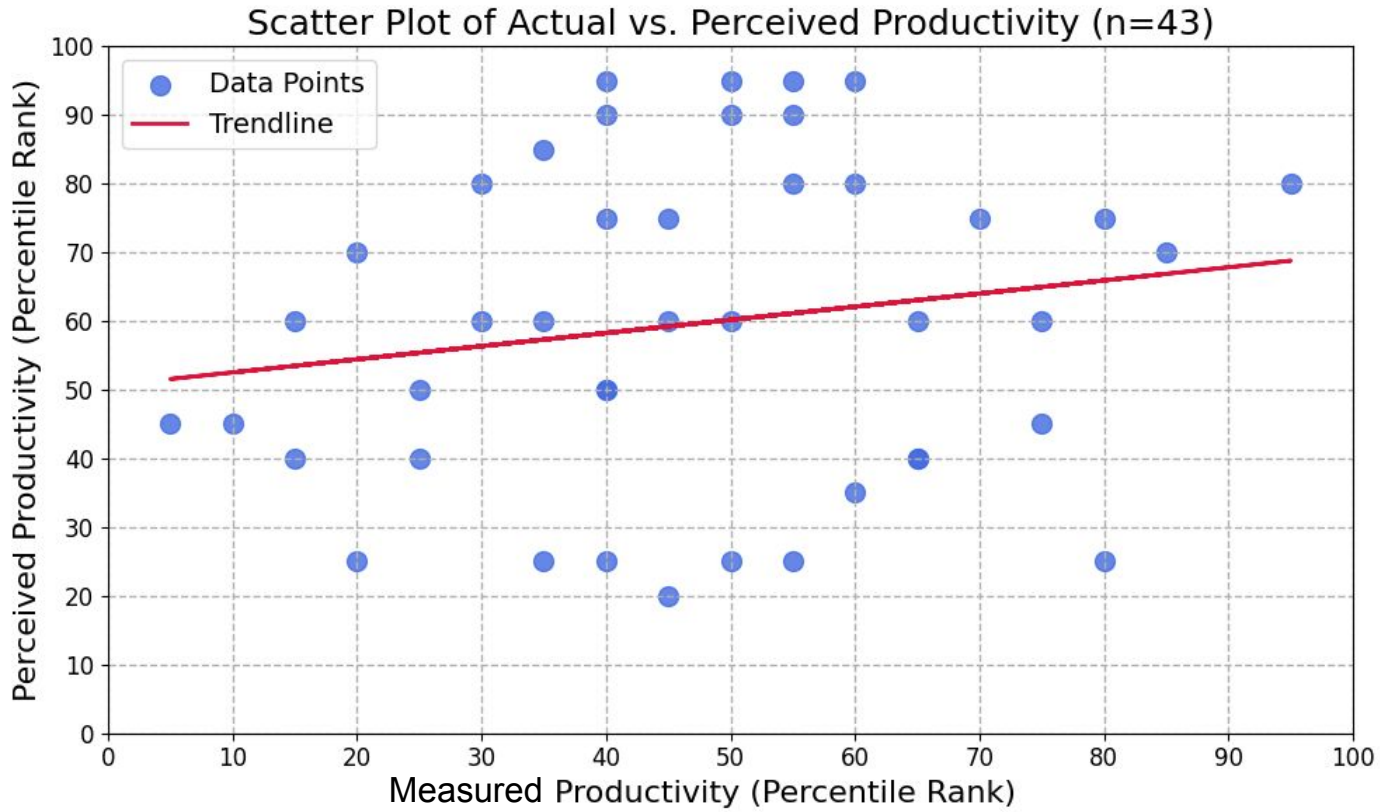
	Problem	Counterproductive Incentive
Lines of Code	More lines != more productivity Not comparable across languages	Encourages verbose, redundant code
# of Commits **# of Pull Requests**	More commits & PRs != more productivity	Encourages artificially small commits/PRs

Story Points are subjective and also don’t measure productivity

Story Points

Problem	Counterproductive Incentive
Subjective & not comparable across teams More story points =/= more productivity	Encourages inflating the number of points a task will require to complete

Self-assessment surveys are an inaccurate way to measure developer productivity



0.17
Correlation (r)

0.03
R^2

Self-assessment surveys (perceived productivity) are an ineffective predictor of productivity

People misjudge their productivity by ~30 percentile points

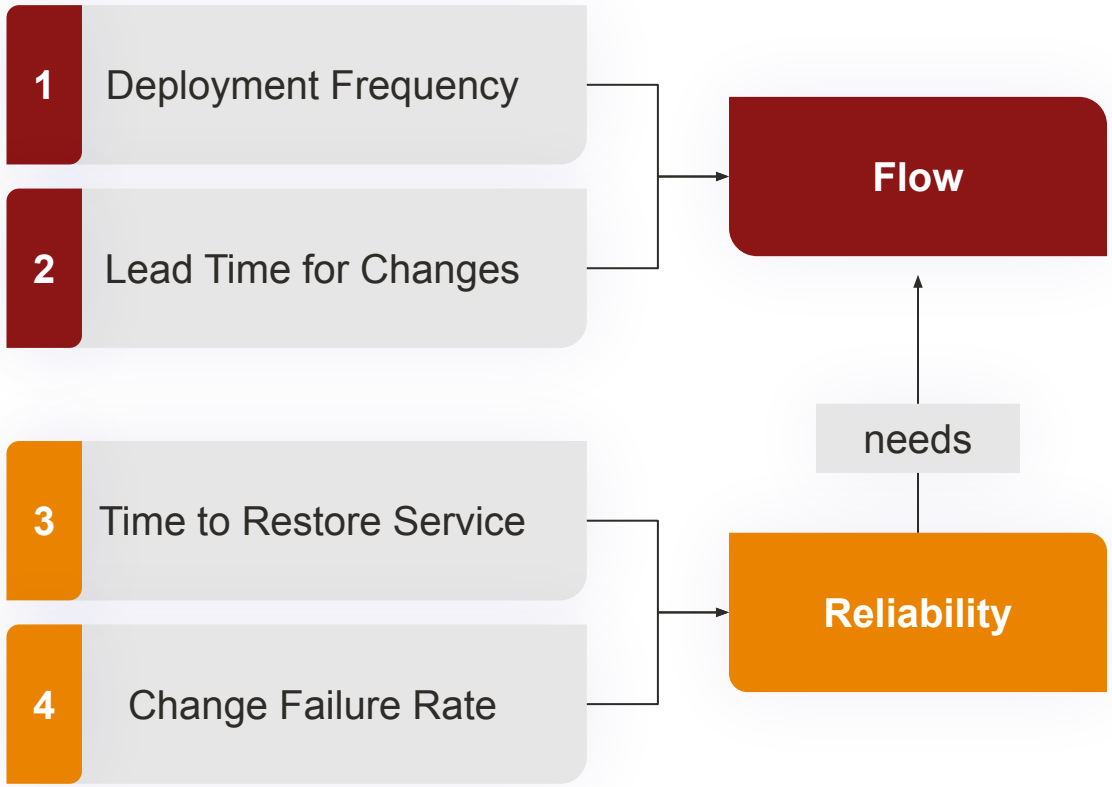
Only **1 in 3** people estimated their productivity **within one quartile**

Surveys are valuable for understanding employee satisfaction and morale

We surveyed 43 software engineers from a statistically representative sample, asking them to rate their productivity on a scale from 0 to 100 in 5-percentile increments, relative to the global average over the past year. We then compared these self-assessments with their actual performance, recorded over the same period, and rounded to the nearest 5 percentile.

DORA Metrics don’t measure productivity, they measure DevOps performance

The 4 DORA Metrics:



Problems with using DORA Flow metrics as a measure of productivity:

1. **Deployment sizes aren't constant within & across teams**
2. **The Flow metrics are gameable**

What a good metric might facilitate

Better Management Decisions

Helps prevent:

- Misguided decisions
- Wasted resources
- Project delays

Fosters Innovation

Good metrics don't get in the way of innovation

Motivates Top Performers

Recognizing hard work and innovation keeps top performers engaged

The difference between Output and Outcomes in Software Engineering

Output

Tangible work produced by engineers

Velocity, building things right

Outcomes

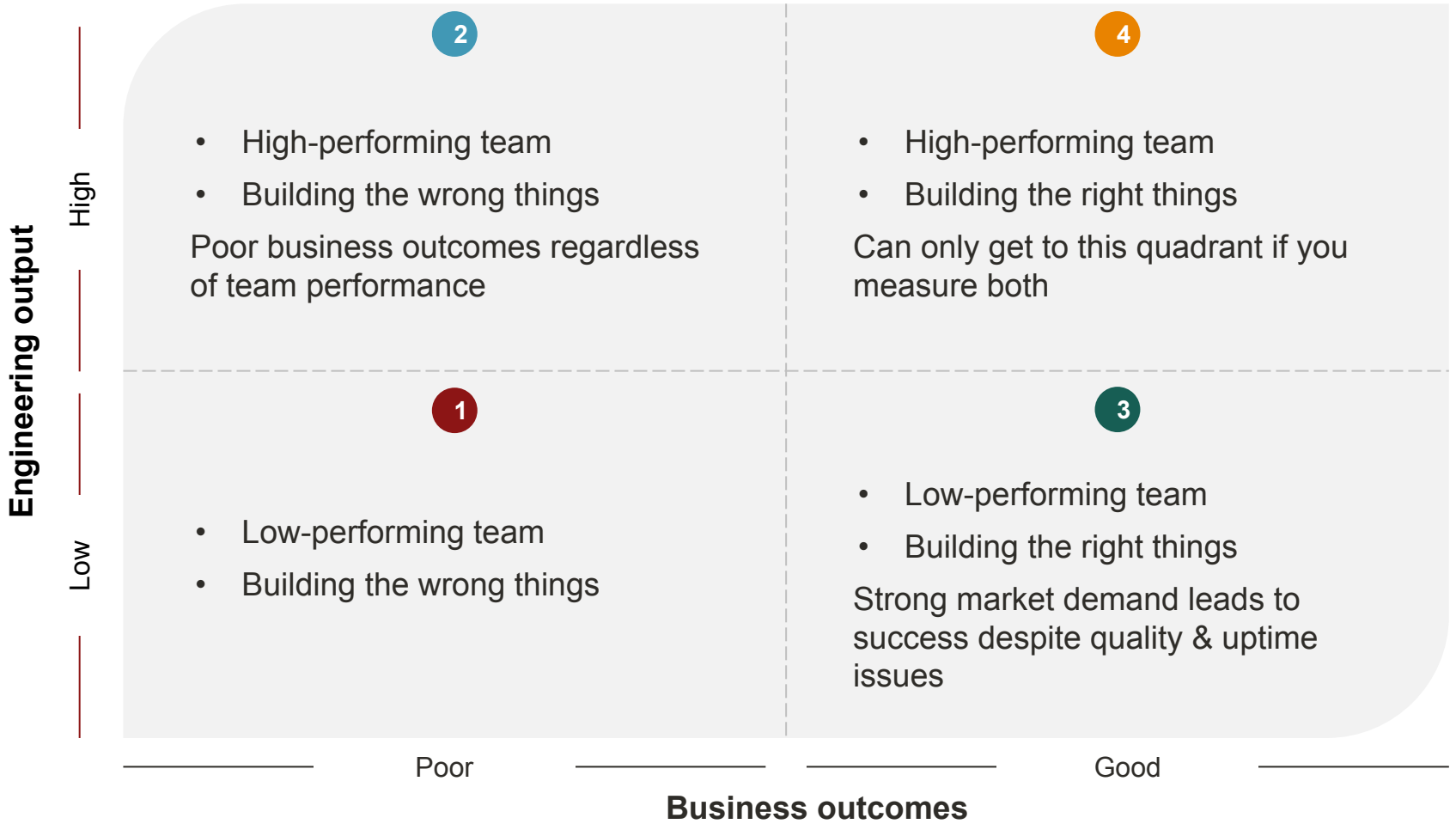
Business results that stem from building the right things

Feature prioritization

Our research focuses on output:

1. Easier to gather objective & comparable data across orgs
2. Product prioritization frameworks exist to drive “building the right things”
3. All else equal, high output is better than low output

Measuring both output and outcomes is necessary to achieve a high-performing software org



1. Problematic
2. High output alone doesn't guarantee success; it needs to be aligned with building the right things
3. Room for improvement in execution
4. Best teams optimize both output and outcomes

Need to measure both output & outcomes

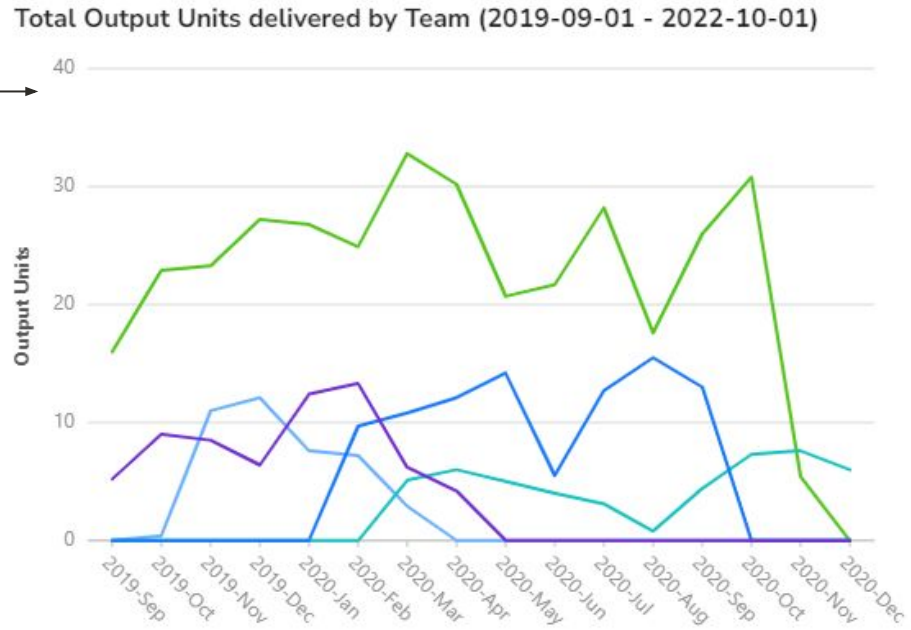
Our model quantitatively evaluates software engineering output by analyzing source code changes on a per-commit basis

How does the algorithmic model work?

1. **Git Repository**
2. **Algorithmic Model**
3. **Output Dashboard**

- **A** — **Analyzes the source code** changes of every commit
- **B** — Quantifies changes based on key dimensions
- **C** — Combines with Git metadata to form a **productivity metric**

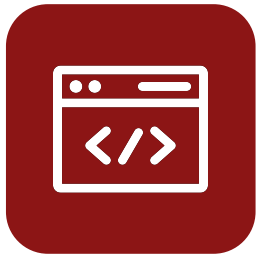
Cohesion	Complexity	Coupling
Data Structures	Interfaces	Methods
Persistence Layers	APIs Consumed	Architectural Patterns
Dependencies	Dependency Injections	…and more



10+ Supported Languages / Frameworks

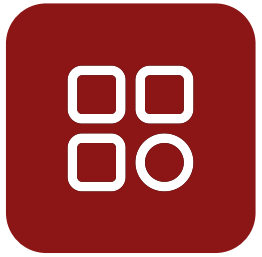
Our current dataset

50,000+ engineers



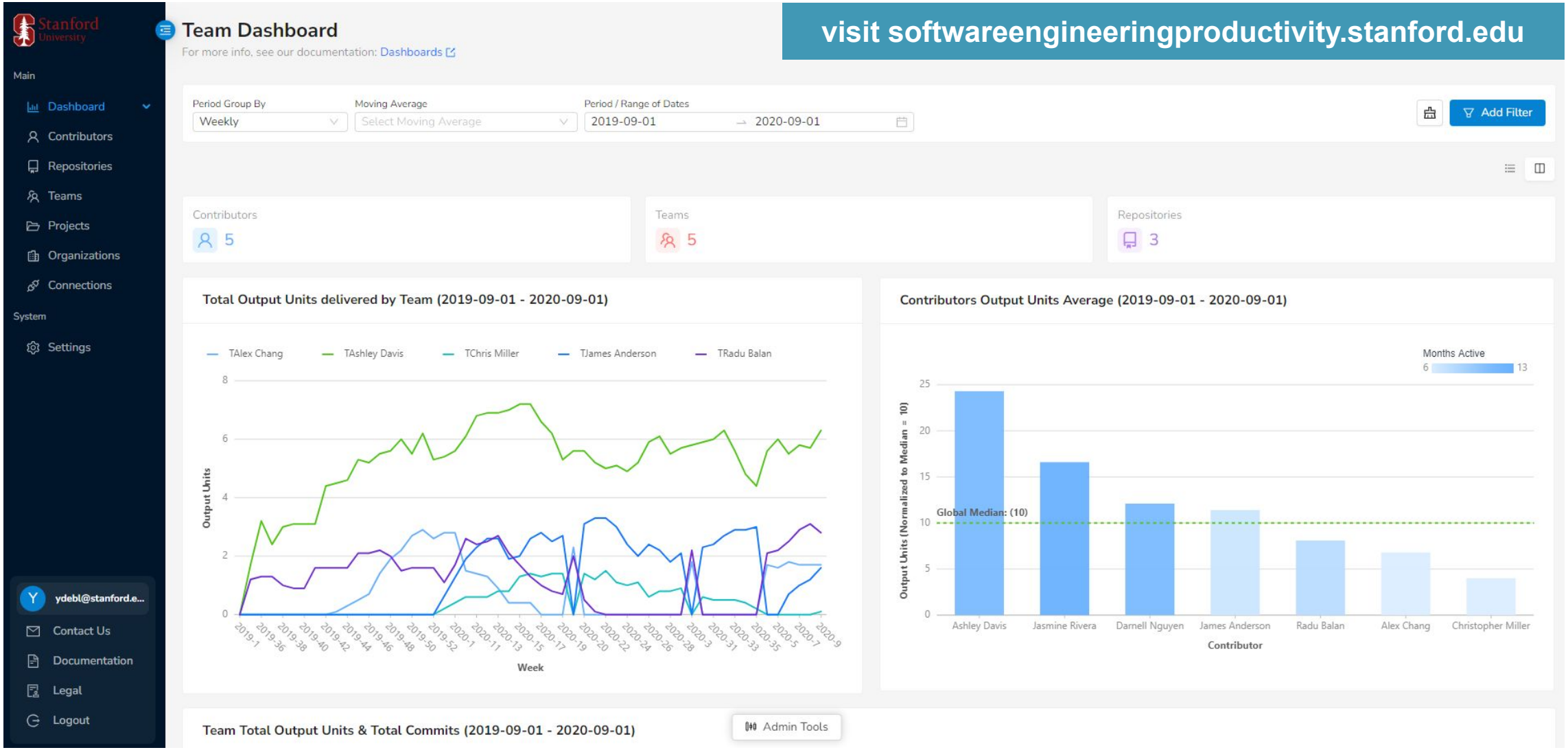
Millions of commits

~2B Lines of Code

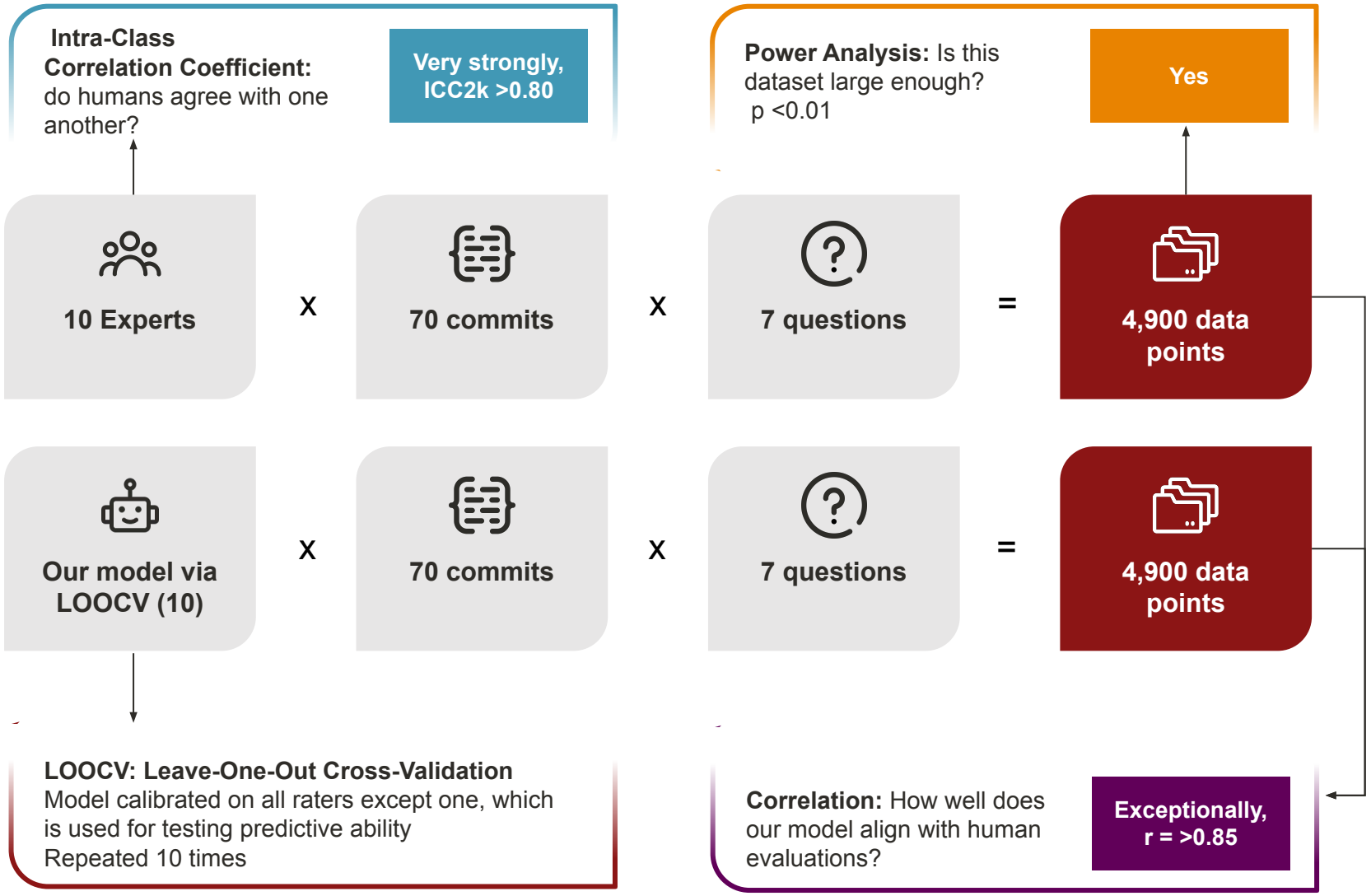


80% private repos

Our research portal provides insights to research participants



How did we test the accuracy of our model?



Our metric (Output Units) doesn't always align with traditional metrics

Author	Output Units 2023 August	Commits 2023 August	Lines of Code 2023 August	Story Points 2023 August
[redacted]	10	21	2,030	8
[redacted]	10	12	2,583	2
[redacted]	10	6	911	3
[redacted]	10	26	4,083	5

Validated Accuracy

~0.85 ICC2k

High Correlation w/ Expert Evaluations

r = >0.80

Fast Commit Processing

<1 second

Scalable Across Orgs.

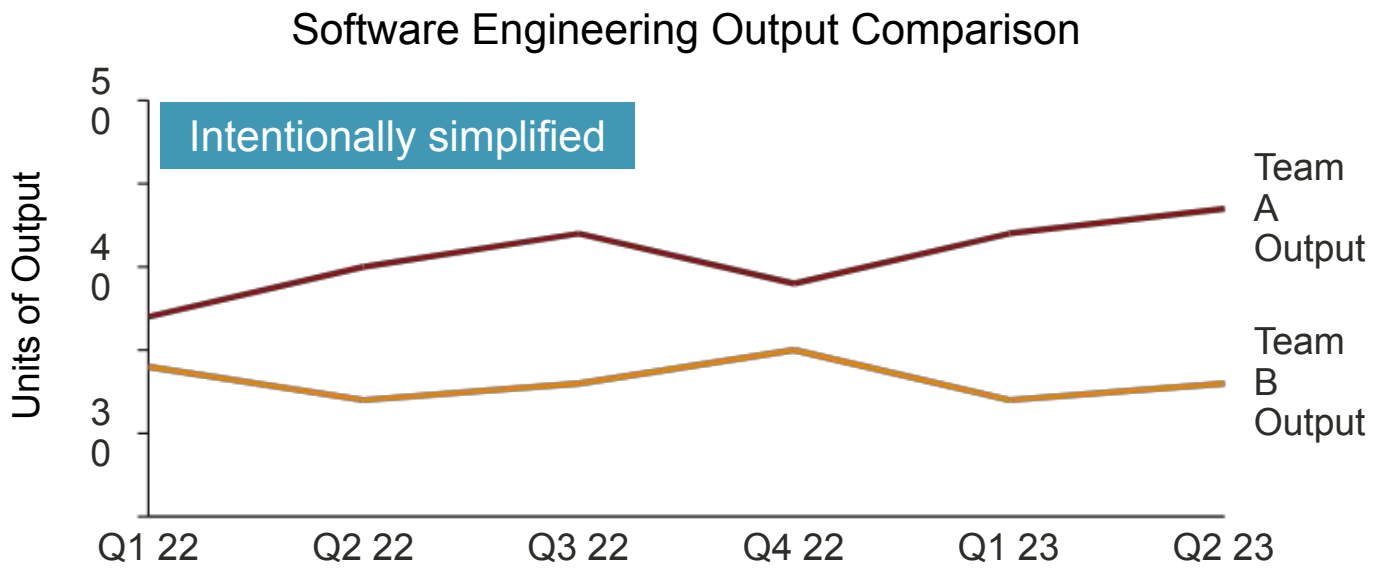
Easy to set up & participate in research

Improvement over traditional metrics

Reads the source code

Case Study 1: Use internal Benchmarking to understand team level differences and best practices

Although Team A delivers more output, Team B is ~30% more 'cost efficient'

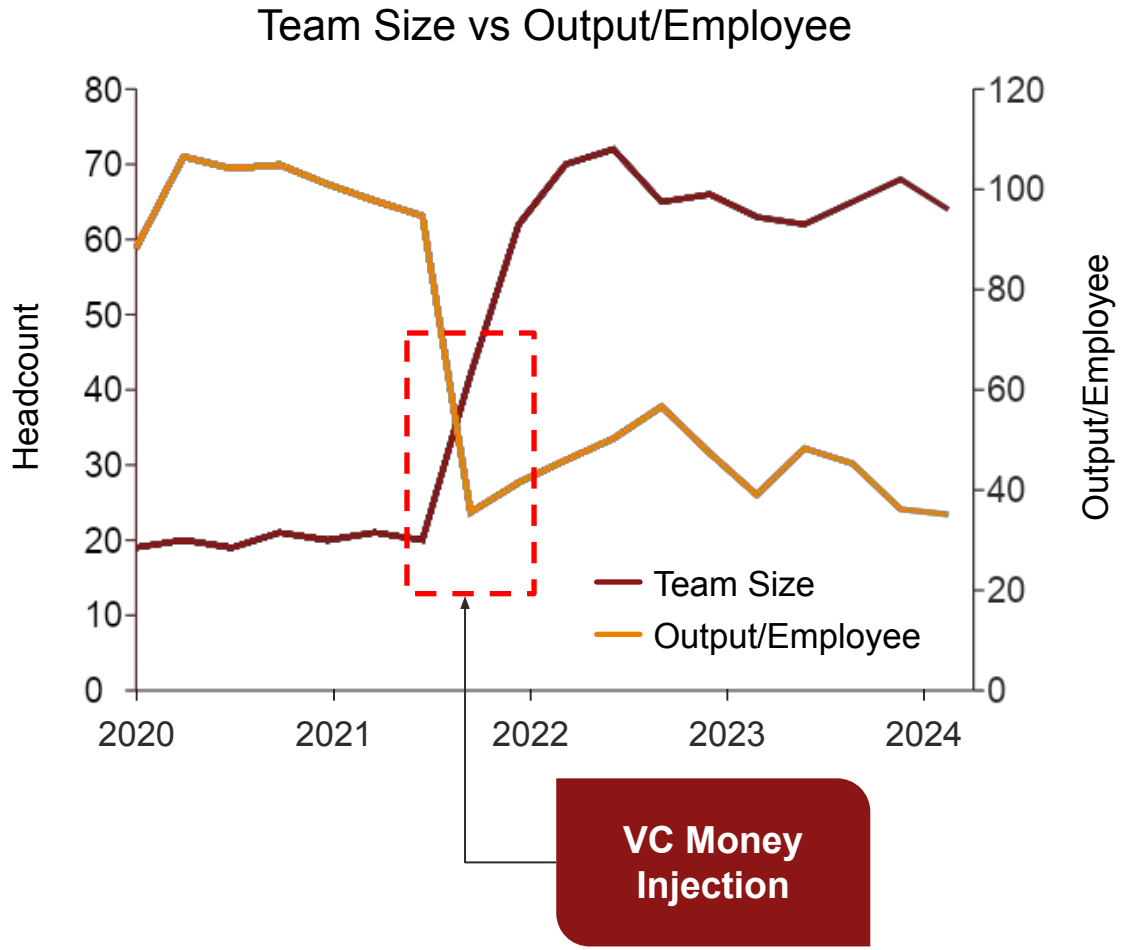


	Team A	Team B	Delta
Avg. Output	41	33.2	-20%
Cost ($K)	1,550	890	-40%
Cost/ Output	37.8	26.8	-30%

Team B is 30% more 'cost efficient'

Our Metric + Other Data = Deeper Insights

Case Study 2: When team size tripled due to VC money, output/employee decreased sharply



Period	Pre-VC Money	Post-VC Money	Growth Factor (Approx.)
Avg. Team Size	20	66	3.3x
Avg. Cost	2,200	6,980	3.2x
Avg. Output	2,010	2,930	1.5x

Team Size & Cost increased by 3x

Output increased by 1.5x

Brooks' Law

Ongoing Research

1. Combination of our measurement with LLMs
2. Cross-language validation of our measurement
3. Comparison to other methods of measuring productivity
4. Self-assessment of productivity does not agree with our measurement
5. Prediction vs sprint outcomes
6. Companies with a higher output achieve better outcomes

Get involved in our research

How can you help:

1. Become an expert rater
2. Join the research with your org as a participant
3. Give feedback on our latest paper

SCAN ME

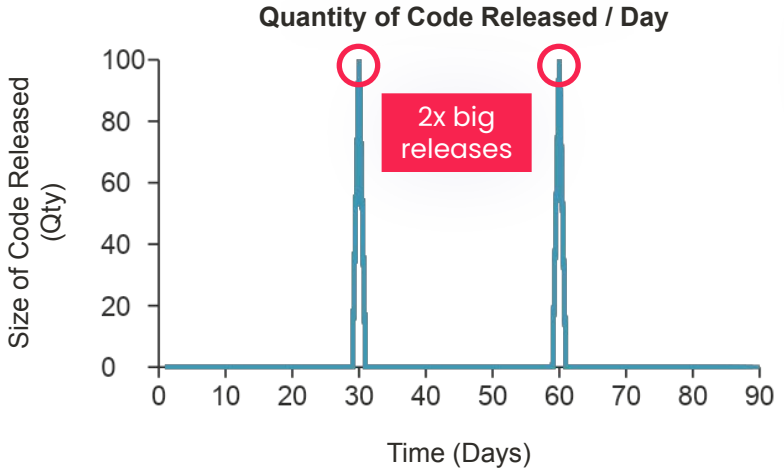
softwareengineeringproductivity.stanford.edu

Backup / Appendix

[1] Deployment Frequency: DORA Metrics Flaw Example (1/2)

Team A

- Building an **iOS app**
- Releases a new version on the App Store 2x a Quarter
- **Releasing a new version more frequently is not possible:**
  - Daily updates would annoy users
  - Takes time to get AppStore approval



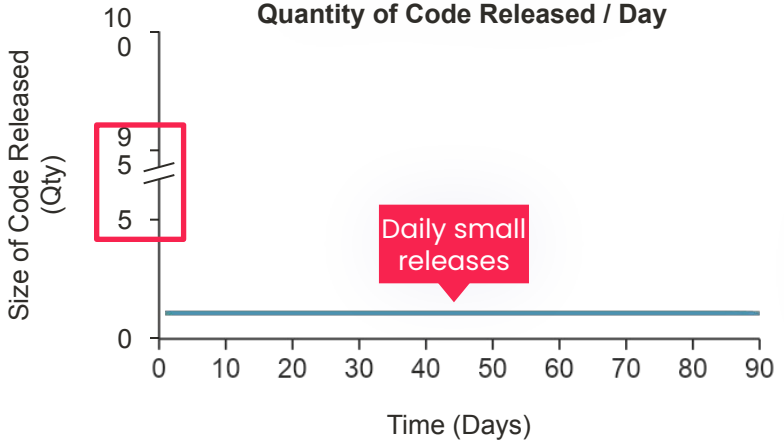
Performance According to DORA Metrics	Medium (Bottom 30%)

# of Code Releases		Size of Each Release		Total Qty Released
2	x	100	=	200

According to DORA, Team A is Bottom 30% yet delivers >2x more code than Team B, who is Top 30%

Team B

- Building a **website-based service**
- Can **release a new version multiple times a day**
  - Users get latest version when they refresh the page
  - No AppStore approval



# of Code Releases		Size of Each Release		Total Qty Released
90	x	1	=	90

Performance According to DORA Metrics	Elite (Top 30%)

[1] Deployment Frequency: DORA Metrics Flaw Example (2/2)

Team A

- Releases code **2 times a day**
- Releases **1 size unit** of code every release

Performance According to DORA Metrics	Elite (Top 30%)

# of Code Releases / Day		Size of Each Release		Total Qty Released / Day
2	X	1	=	2

According to DORA, both Teams are Elite, yet Team B delivers 5x more code than Team A

Team B

- Also releases code **2 times a day**
- Releases **5 size units** of code every release

Performance According to DORA Metrics	Elite (Top 30%)

# of Code Releases / Day		Size of Each Release		Total Qty Released / Day
2	X	5	=	10

[2]Lead time for Changes: DORA Metrics Flaw Example (1/2)

Team A

- Building an **iOS app**
- Releases a new version on the App Store 2x a Quarter
- **Releasing a new version more frequently is not possible:**
  - Daily updates would annoy users
  - Takes time to get AppStore approval

Time Between Releases	30 days

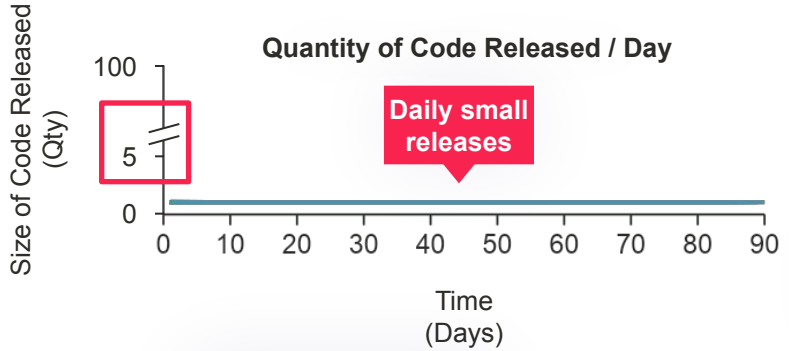
Performance According to DORA Metrics	Medium (Bottom 30%)

# of Code Releases		Size of Each Release		Total Qty Released
2	x	100	=	200

According to DORA, Team A is Bottom 30% yet delivers >2x more code than Team B, who is Top 30%

Team B

- Building a **website-based service**
- Can **release a new version multiple times a day**
  - Users get latest version when they refresh the page
  - No AppStore approval



Time Between Releases	<1 day

# of Code Releases		Size of Each Release		Total Qty Released
90	x	1	=	90

Performance According to DORA Metrics	Elite (Top 30%)

[2]Lead time for Changes: DORA Metrics Flaw Example (2/2)

Team A

- Releases code **2 times a day**
- Releases **1 size unit** of code every time

Performance According to DORA Metrics	Elite (Top 30%)
Time Between Releases	<1 day

# of Code Releases / Day		Size of Each Release		Total Qty Released / Day
2	x	1	=	2

According to DORA, both Teams are Elite, yet Team B delivers 5x more code than Team A

Team B

- Also releases code **2 times a day**
- Releases **5 size units** of code every time

Performance According to DORA Metrics	Elite (Top 30%)
Time Between Releases	<1 day

# of Code Releases / Day		Size of Each Release		Total Qty Released / Day
2	x	5	=	10

Why DORA Metric #2, Lead Time for Changes, is Flawed

1	Lead Time for Changes is *ALSO* NOT a measure of output
2	It is *also* a measure of to what degree you've adopted a CI/CD (Continuous Integration / Continuous Development) way of working
3	CI/CD practices have gained such widespread adoption that it is very easy to rank "Elite" in this metric
4	For companies that can't release frequently (e.g. iOS Apps, Financial Services, etc.) this metric is completely meaningless
5	This metric will become irrelevant very soon

Why DORA Metric #3, Time to Restore Service, is Flawed

Time to Restore Service	When a software outage occurs, how long does it take to restore service?

Software delivery performance metric	Elite	High	Medium	Low
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months

1. This is an almost meaningless metric. When was the last time you ran into a site going down for a week, let alone 6 months?
2. Orgs with teams dispersed across timezones will perform better by default – outages can happen during the middle of the night

Why DORA Metric #4, Change Failure Rate, is Flawed

Change Failure Rate	What % of your software versions have an incident/bug?

Software delivery performance metric	Elite	High	Medium	Low
⚠ Change failure rate	0%-15%	16%-30%	16%-30%	16%-30%

This is not a typo

1. Teams deploying less frequently (and therefore with bigger deploys) will have a higher chance of each deploy being flagged as bugged

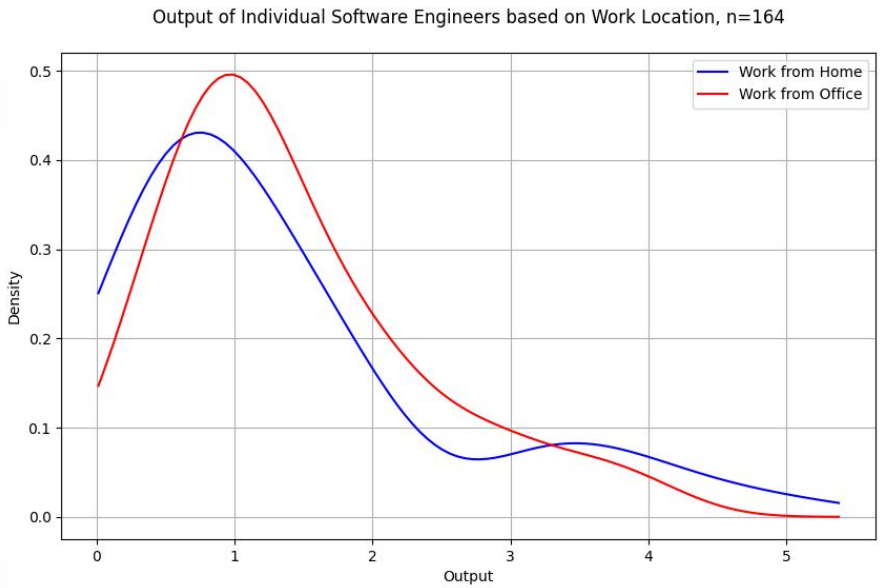
The bottom 25% of software engineers working from home severely underperform, while the top 10% significantly outperform their office-based counterparts

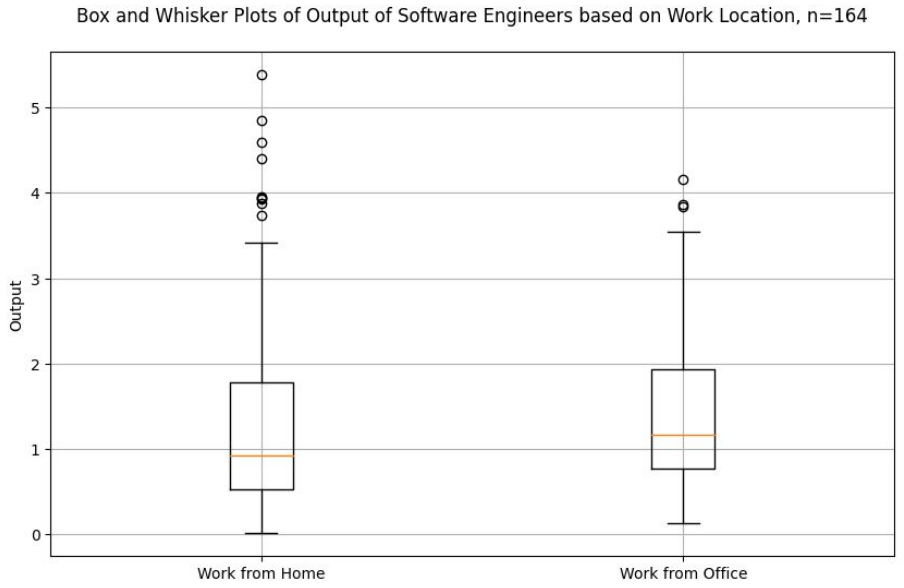
Selective underperformance in remote work

- The **lowest-performing 12%** of engineers who work from home produce **less than 5%** of the output that a median engineer delivers

But also exceptional overperformance

- The **top 16%** of engineers **working remotely** exhibit an output equivalent to or exceeding the **top 5%** of **office-based** engineers





Percentile	Work from Home Output	Work from Office Output	% Difference
5	0.03	0.25	-87.9%
25	0.52	0.77	-32.6%
50	0.92	1.17	-21.3%
75	1.78	1.94	-7.9%
95	3.94	3.44	+14.5%

Quartile	Work from Home Output	Work from Office Output	% Difference
Q1 (0-25)	0.26	0.47	-45.12%
Q2 (25-50)	0.72	0.96	-25.1%
Q3 (50-75)	1.32	1.46	-9.6%
Q4 (75-100)	3.09	2.74	+12.9%
Bottom 50%	0.49	0.72	-31.8%
Top 50%	2.24	2.12	+5.6%