Abusing Metrics for Dummies

How to manipulate data to make us look good?

Elad Sofer - @eladsof

elad@practical-agile.com

Practical Agile
Agile in your Context

PRACTICAL
SOFTWARE

Elad Sofer - CV

- Software developer
- LeSS Certified Trainer
- Agile coach
- Organizational coach

Practical Agile

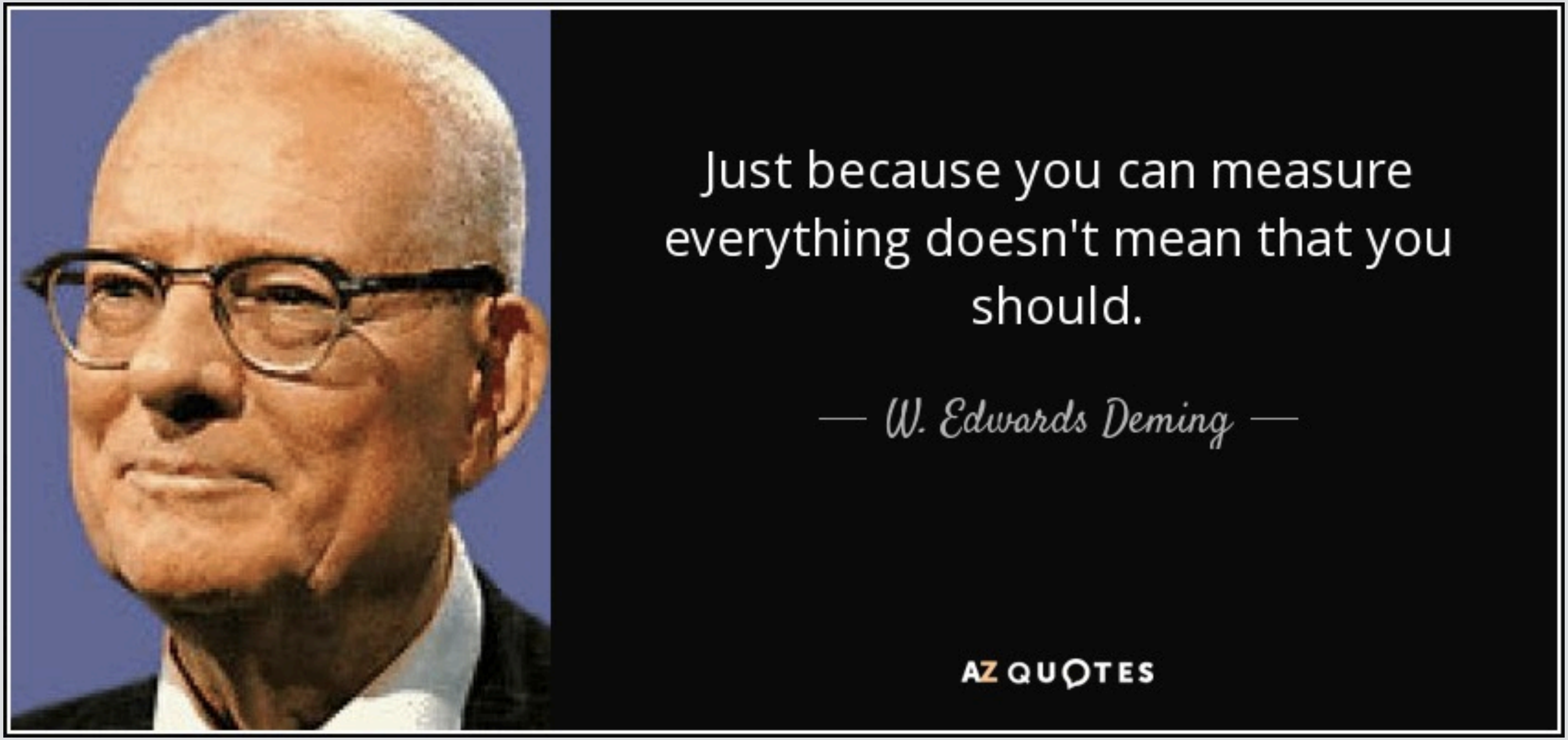
Agile in your Context

PRACTICAL SOFTWARE

Question:

What do you measure?

What are you expecting to achieve?

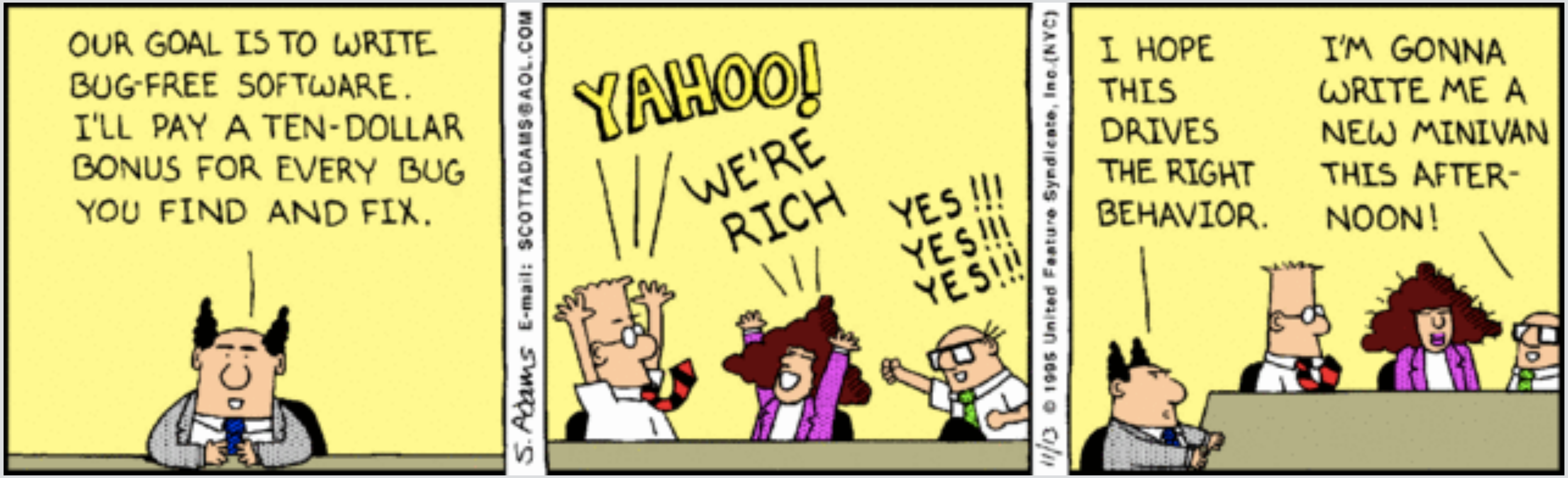


Just because you can measure everything doesn't mean that you should.

— W. Edwards Deming —

The danger of individual metrics

Avoid using metrics as incentives


OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
S. Adams E-mail: SCOTTADAMS@AOL.COM
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
11/13 © 1995 United Feature Syndicate, Inc.(NYC)
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTER-NOON!

The observer effect

The act of **observing** something **changes** it

*Not necessarily in the way you expect it to change

Avoid using metrics for:

- Manipulating behavior
- Manipulating management
- Making yourself feel good

Manipulative metrics example

- Unit test coverage %
    - We want the team to increase quality.
- Velocity as a productivity metric
    - We want the teams to developer faster.
- Sprint score (the new “planned vs actual”)
    - We want to be more predictable

Political metrics example

- Transformation % progress
  - We want management to see that we are implementing the process changes.
- Cost of bugs
  - We want to get approval for this new design which we think is better.
- Time spent waiting for <whatever>
  - We want to find someone to blame so that you leave us alone.

Vanity metrics example

- # Items that are in “dev done” status
    - We want to show that we are a “good" dev team.
- # Page views
    - We want to show as if our marketing efforts are paying off.
- Employee churn rate
    - We want to show that we are a great place to work for.

Metrics are not a replacement for leadership

“Don’t look with your eyes, look with your feet… A person who looks only at the numbers is the worst of all.”

Taiichi Ohno

As a team..

Don’t focus on the metric and the numbers.

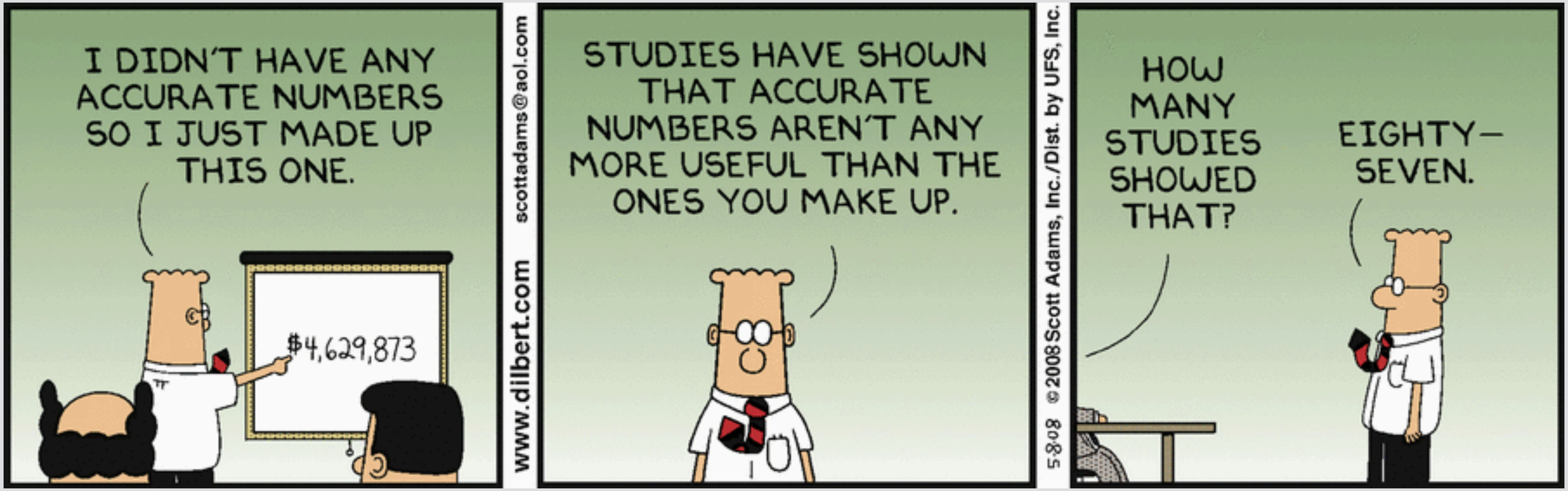
Focus on who sets them and how they intend to use them.

Metrics are not goals

- Metrics are tools to get visibility towards achieving a goal.
- Avoid setting “top-down” metrics which you cannot directly impact.
- Replace metrics with business goals and help the teams choose the right metrics.
- **“Because I want to know” is not a goal!**

How to metrics help achieving goals?

They enable **better and faster** decision making


I DIDN'T HAVE ANY ACCURATE NUMBERS SO I JUST MADE UP THIS ONE.
$4,629,873
www.dilbert.com
scottadams@aol.com
STUDIES HAVE SHOWN THAT ACCURATE NUMBERS AREN'T ANY MORE USEFUL THAN THE ONES YOU MAKE UP.
5-8-08 © 2008 Scott Adams, Inc./Dist. by UFS, Inc.
HOW MANY STUDIES SHOWED THAT?
EIGHTY-SEVEN.

“Without Data you’re just another person with an opinion”

W. Edwards Deming.

Not Deming

A good metric is:

- Easy to accurately measure.
- Actionable.
- Aligned with a compelling goal.
- Hard to manipulate.
- Owned by the right people.

MONTY PYTHON

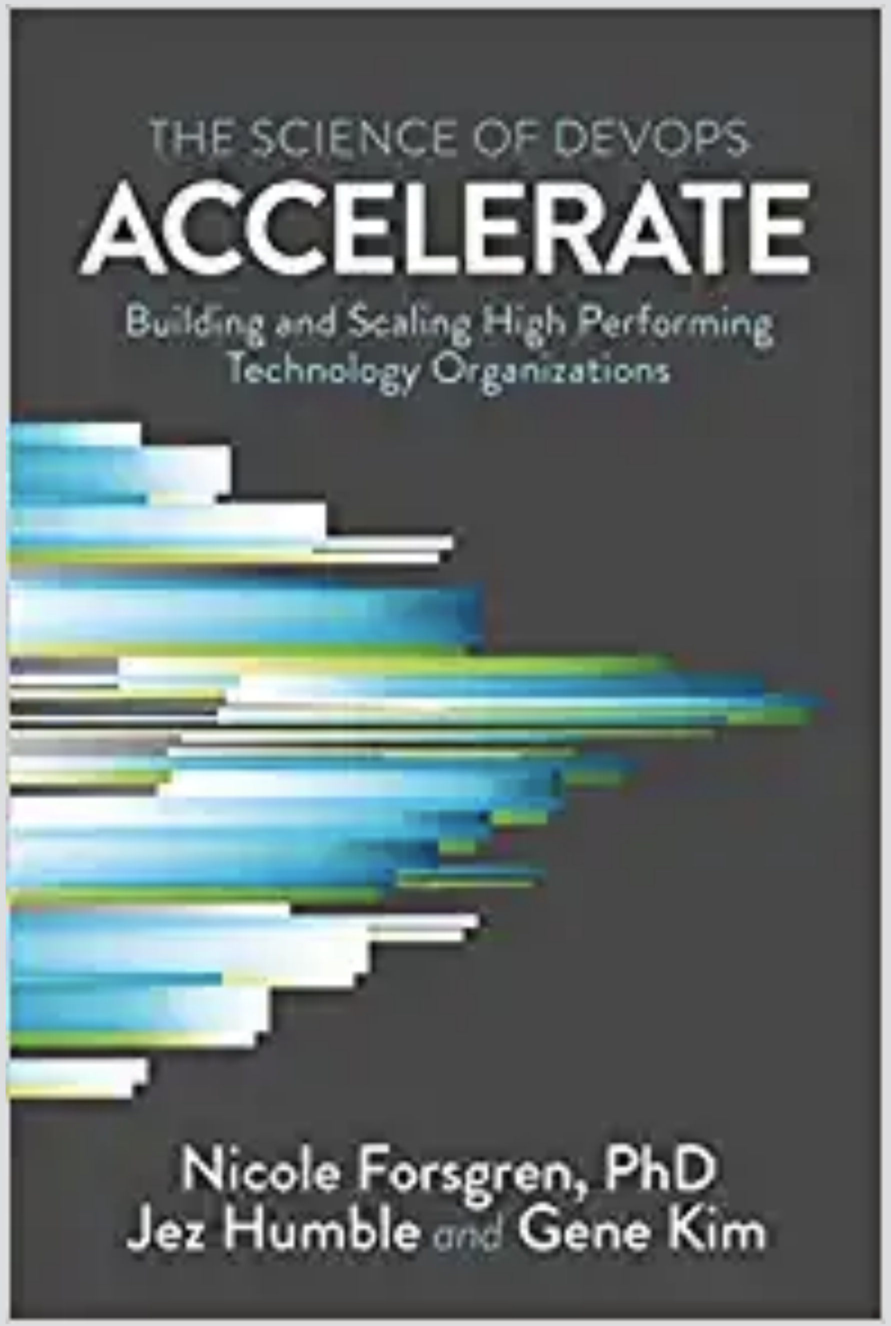
and the Holy Grail

The bad news

We don’t know how to directly measure effectiveness of software development teams.

The good news

Software development effectiveness can be **indicated** by a combination of **4 metrics**



The 4 metrics that count

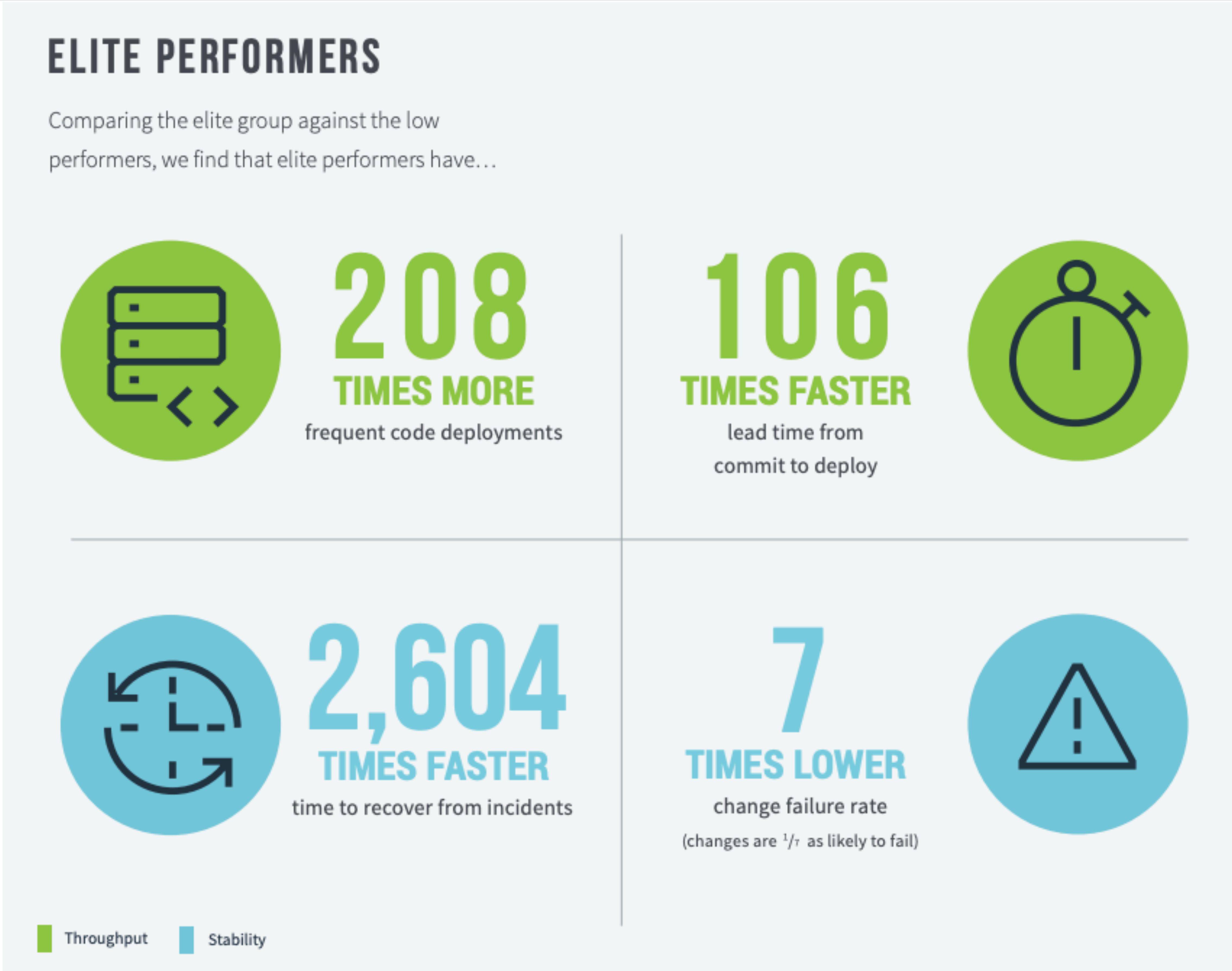
- **Deployment frequency**
    - how often does your organization deploy code to production?
- **Lead time for changes**
    - how long does it take to go from code committed to code successfully running in production
- **Time to restore service**
    - how long does it take to restore service when a service incident that impacts users occurs
- **Changes failure rate**
    - what percentage of changes to production or released to users result in degraded service

Comparing organization

Organiztion effectiveness	Elite	High	Medium	Low
Deployment frequency	On demand	One per day - One per week	once per week - once per month	Less than once per month
Lead time for changes	Less Than a day	One day - one week	One week - one month	More than a month
Time to restore service	Less than one hour	Less than one day	Less than one day	Between one week and one month
Change failure rate	0-15%	0-15%	0-15%	40-60%

Comparison is based on : Profitability, market share, customer satisfaction, Number of customers

*Source: State of devops 2019 report



ELITE PERFORMERS

Comparing the elite group against the low performers, we find that elite performers have…

208 TIMES MORE frequent code deployments

106 TIMES FASTER lead time from commit to deploy

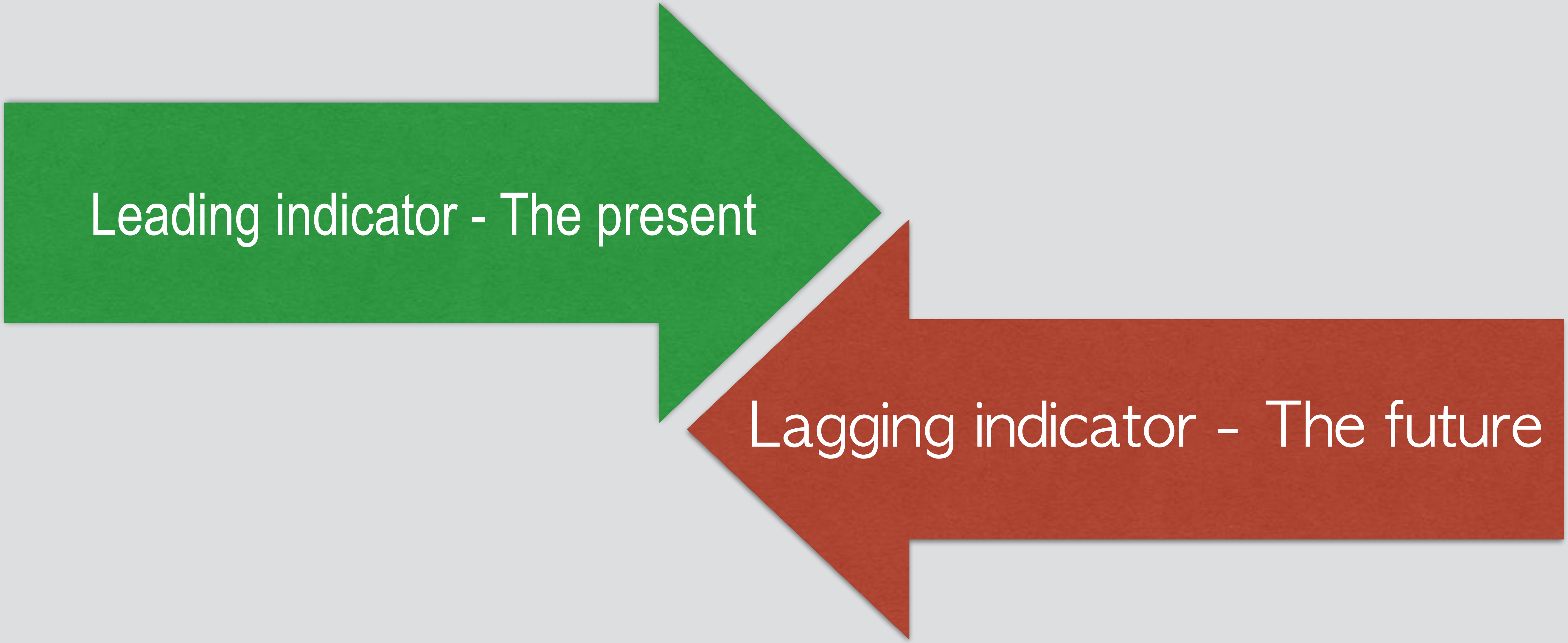
2,604 TIMES FASTER time to recover from incidents

7 TIMES LOWER change failure rate (changes are $^1/_7$ as likely to fail)

🟩 Throughput 🟦 Stability

*Source: State of devops 2019 report

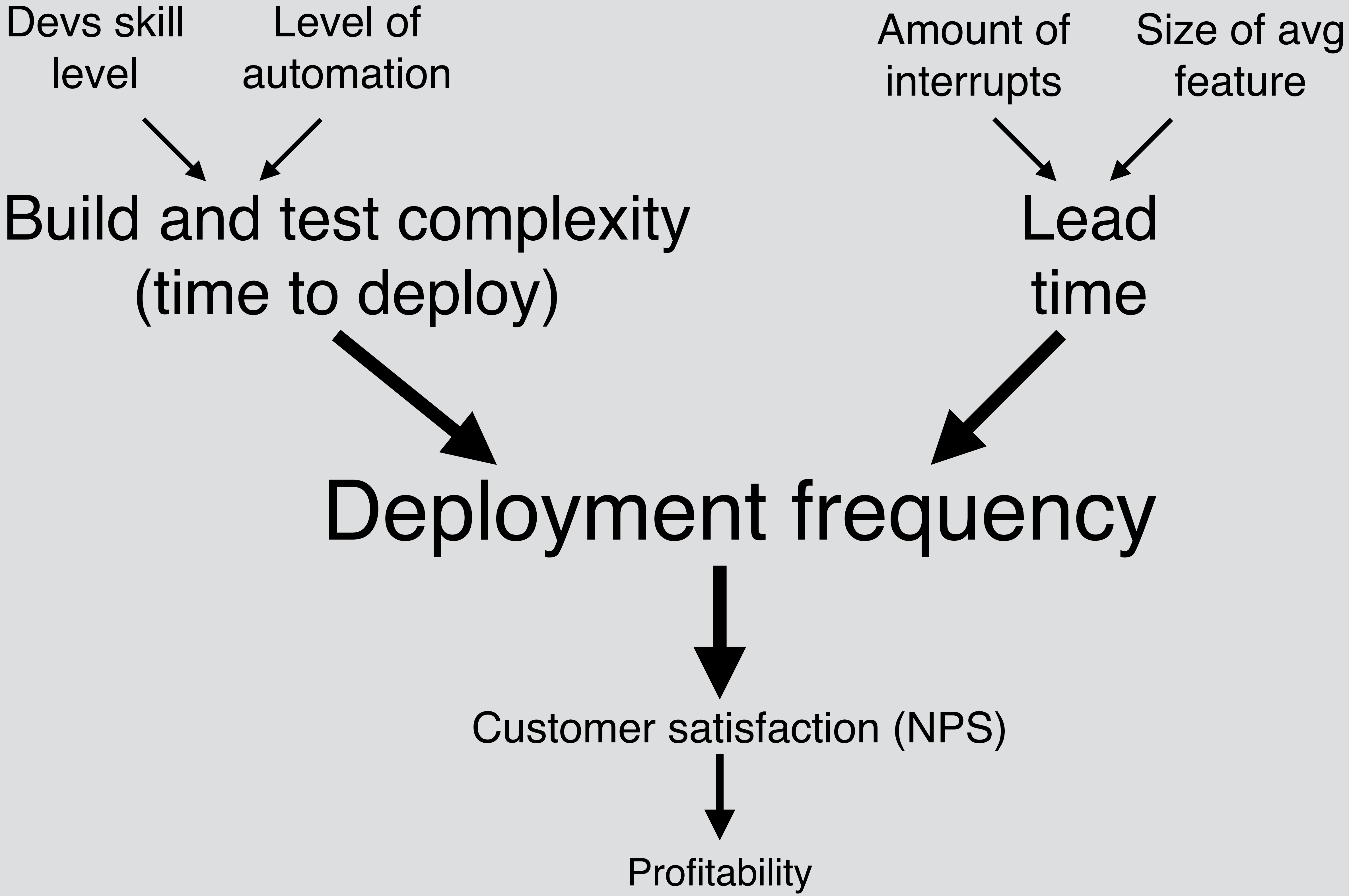
Leading \ Lagging metrics



Leading indicator - The present

Lagging indicator - The future

Is it leading or lagging?



Devs skill level

Level of automation

Amount of interrupts

Size of avg feature

Build and test complexity (time to deploy)

Lead time

Deployment frequency

Customer satisfaction (NPS)

Profitability

Before using a metric - test it

- Define an experiment to test the connection between the leading and the lagging indicator
- The connection may not be as strong as you think
- Focus on learning not on achieving the expected result.

More potentially useful metrics

- Cyclomatic complexity
- eNPS
- Feature usage count
- Psychological Safety
- Level of confidence
- # of teams needed to create value.
- % of mistakes (not what you think…)

Summary

- Most metrics are wrong, some are useful
- People will often misinterpret the metrics.
- Deliberate uncertainty - experiment & Learn.
- Use metrics in pairs (leading -> Lagging).
- Avoid measuring individuals.
- Metrics should be owned by the teams.

Thank you!