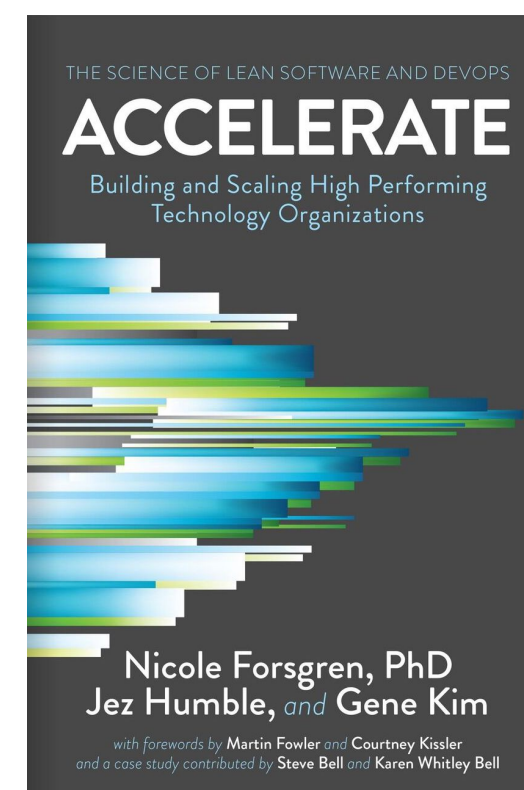




if you don't know where you're going  
it doesn't matter how fast you get there

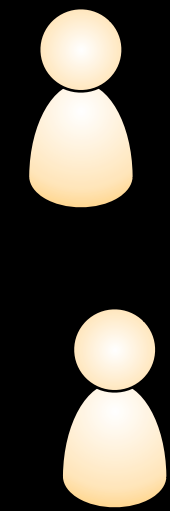
Nicole Forsgren, PhD @nicolefv  
Jez Humble @jezhumble  
#qconf 2018



# "the enterprise"

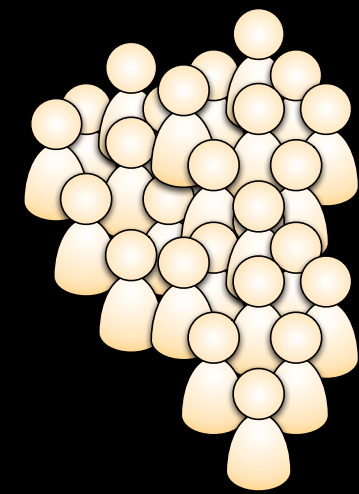
Ping!

Business

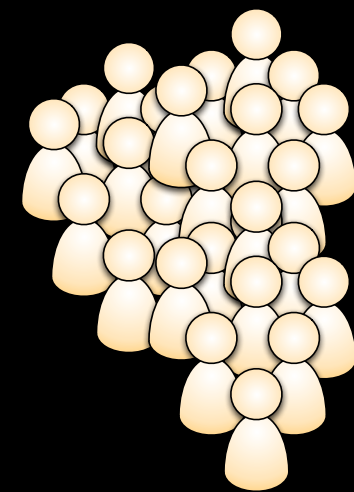
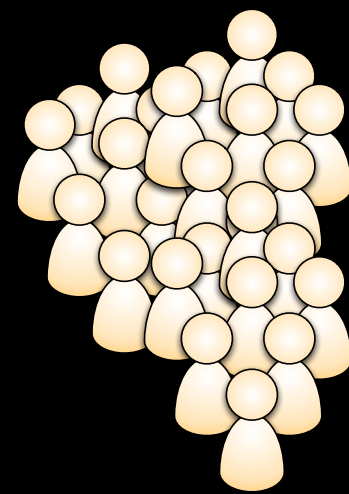


Engineering

Project C

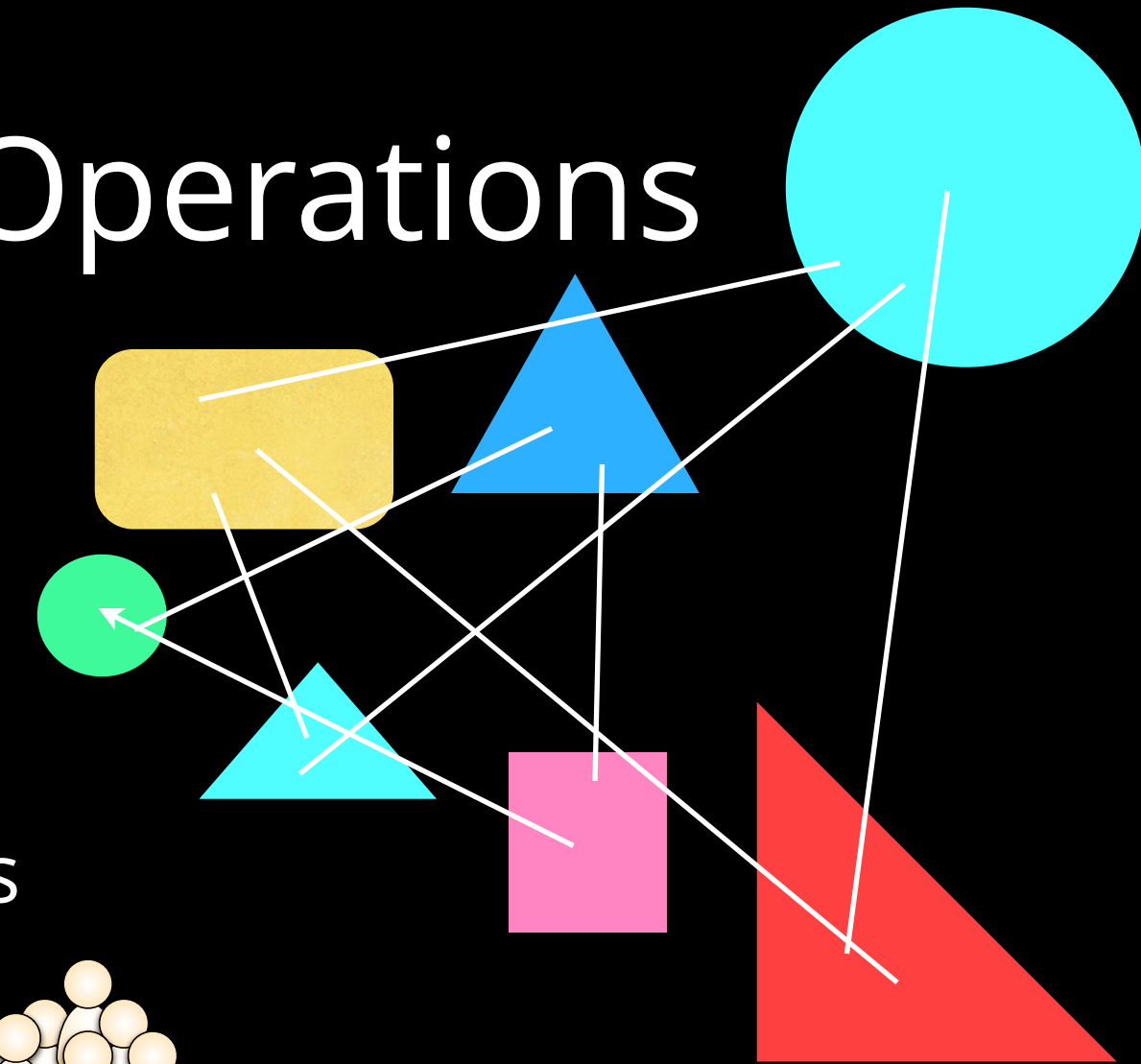


Project A

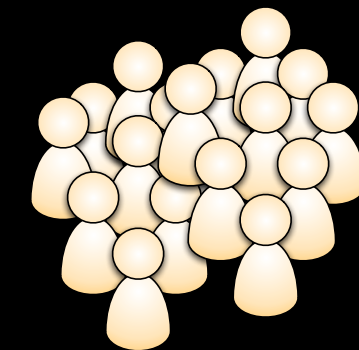


Project B

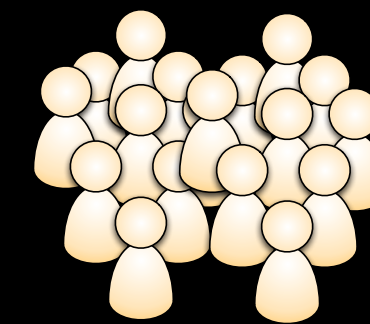
Operations



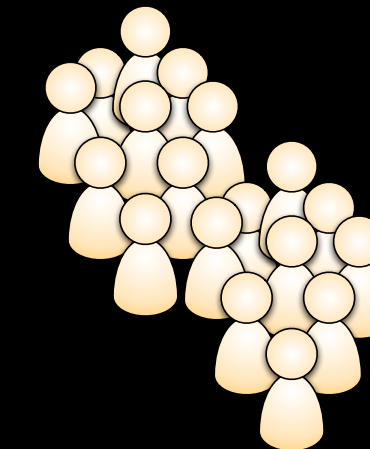
DBAs



Service desk



Infrastructure team



Value stream

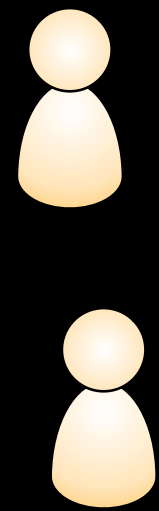


# enterprise projects

Ping!

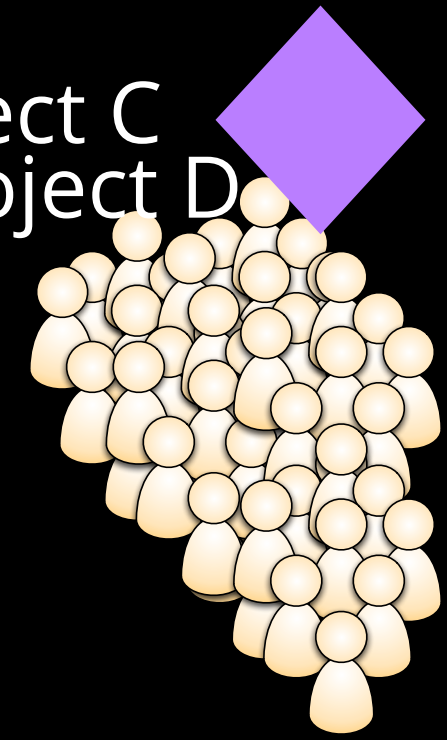
## Business

Let's create a new product

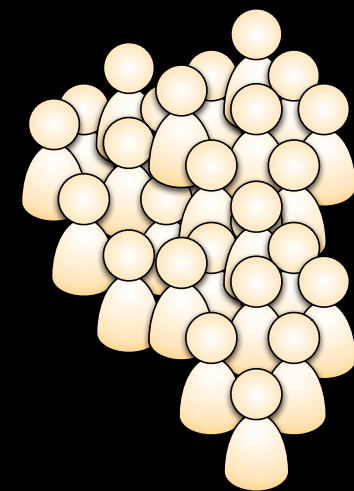
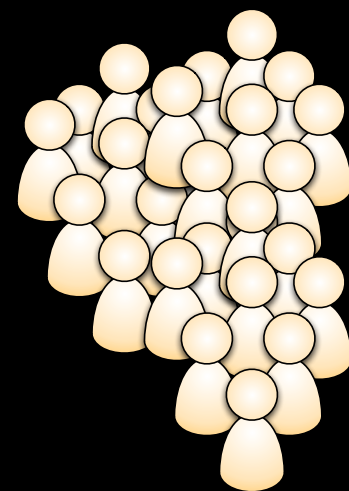


## Engineering

Project C  
Project D

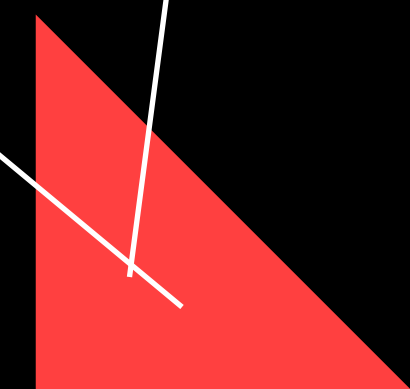
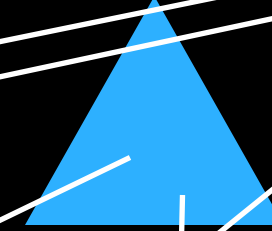
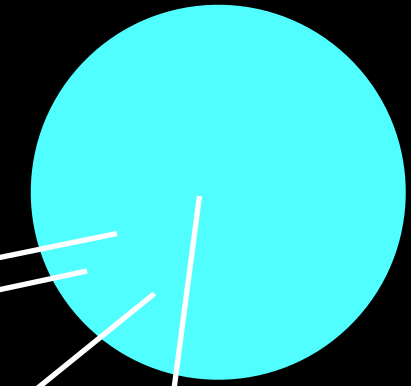


Project A

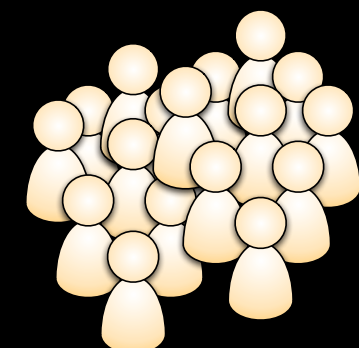


Project B

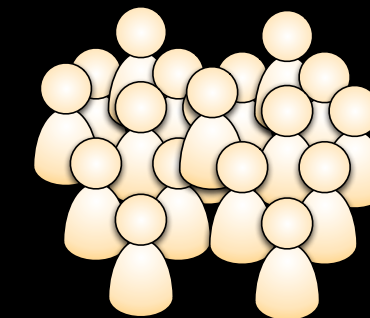
## Operations



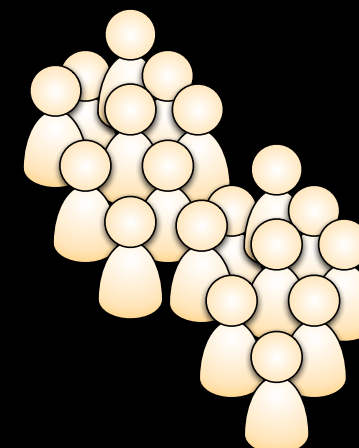
DBAs



Service desk



Infrastructure team

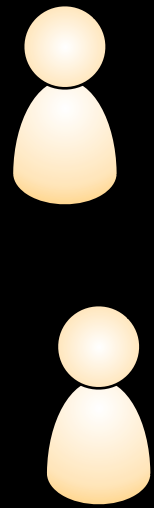


Value stream



Oh no!

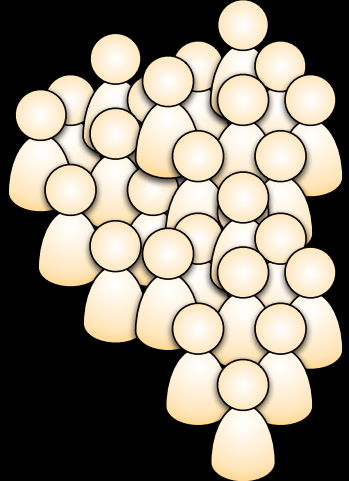
# Business



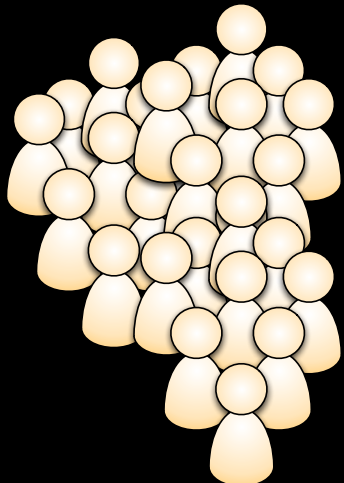
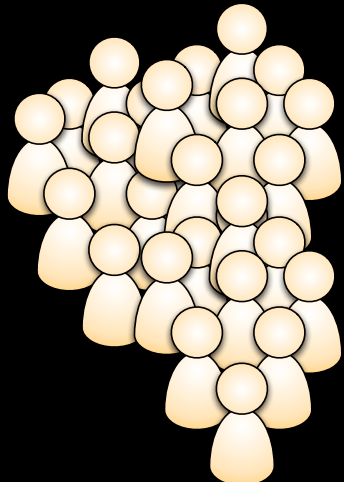
We're going agile!

# Engineering

Project D



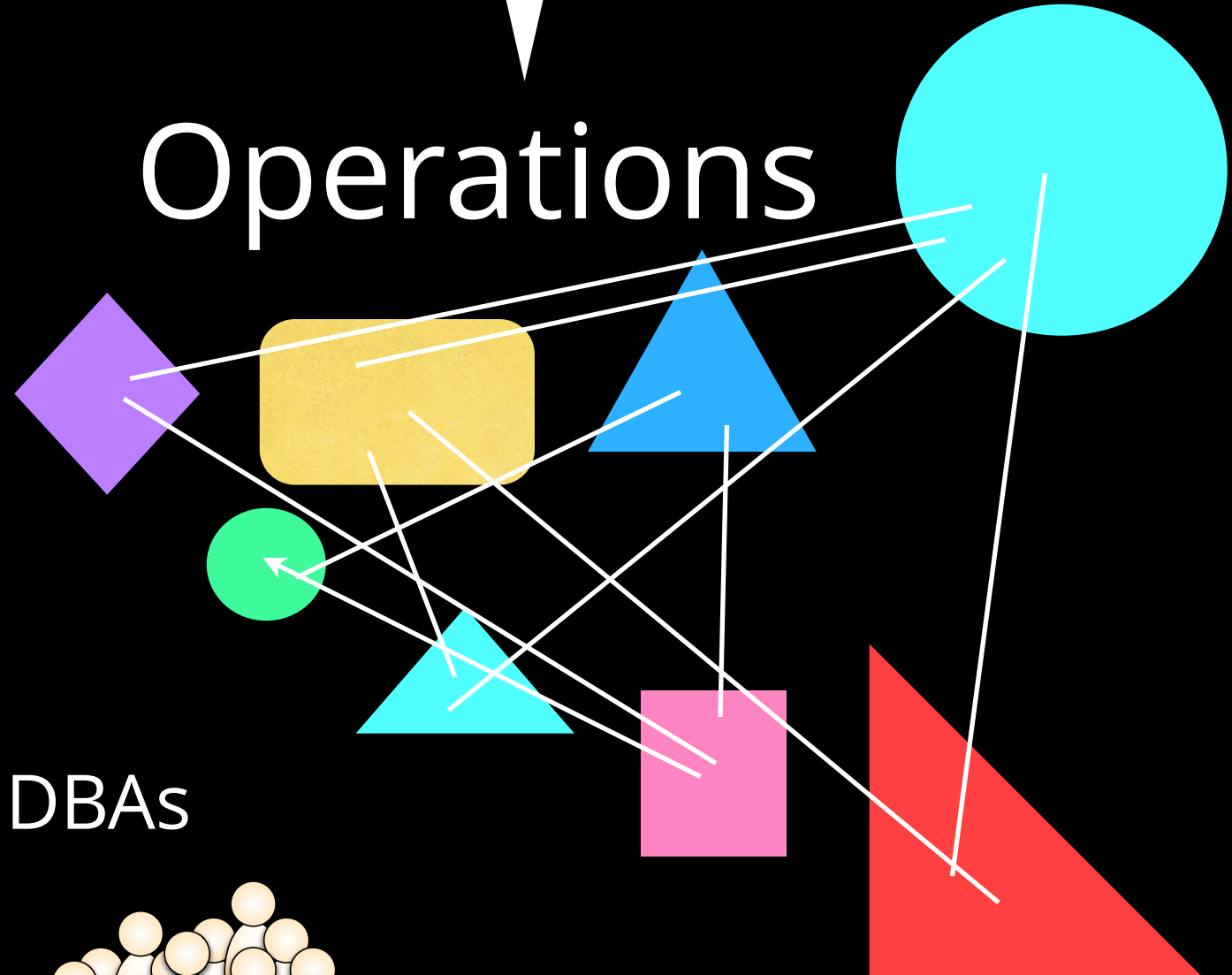
Project A



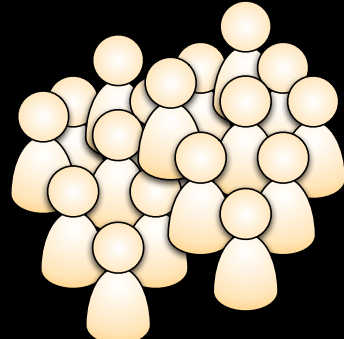
Project B

Oh no!

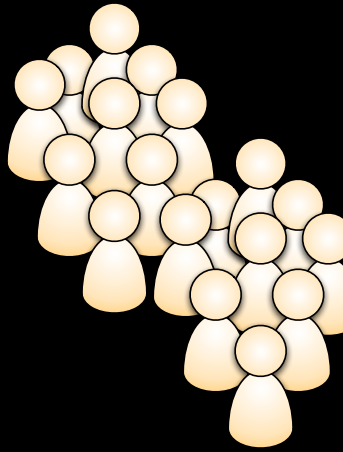
# Operations



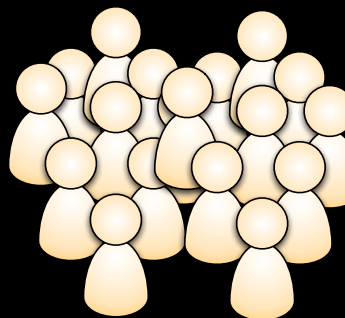
DBAs



Infrastructure team



Service desk



# Value stream



Our test-driven code follows SOLID principles

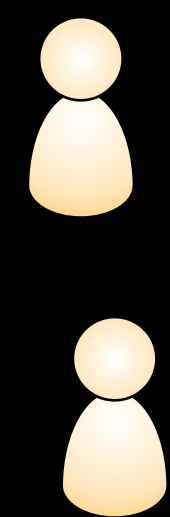
Change management

Shame it doesn't work

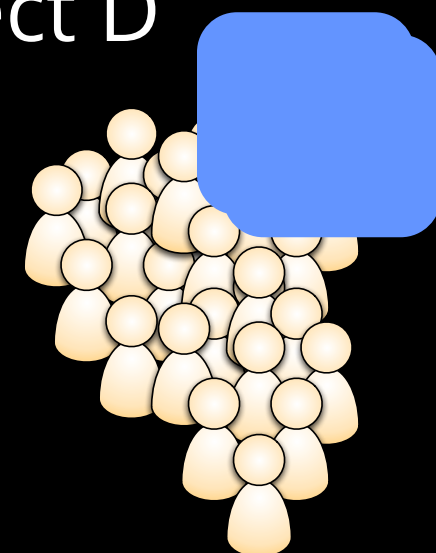
Business

Engineering

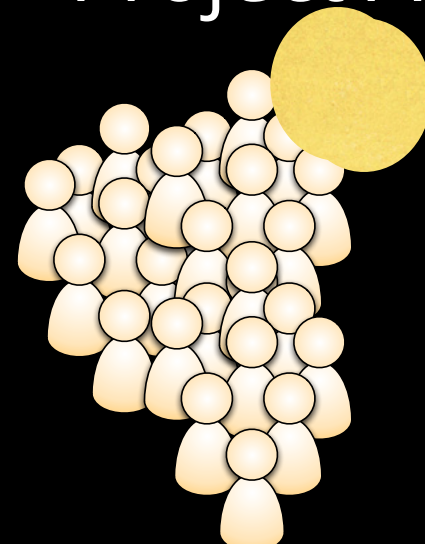
Operations



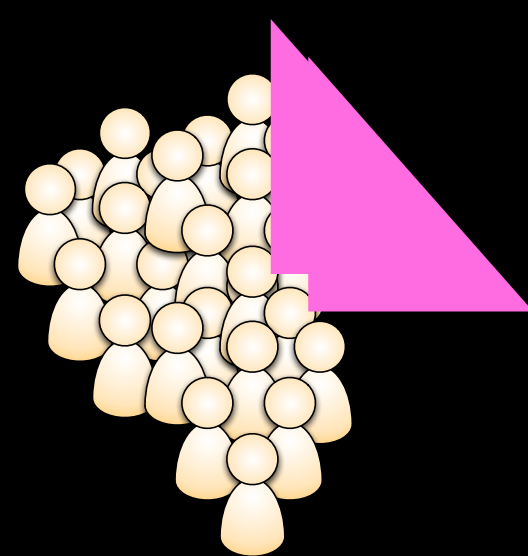
Project D



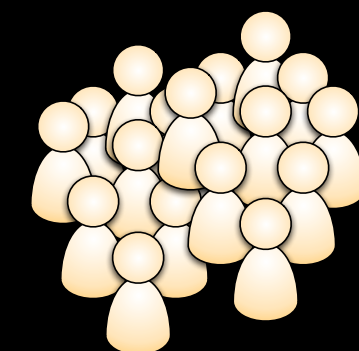
Project A



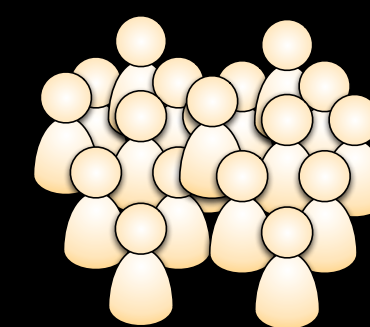
Project B



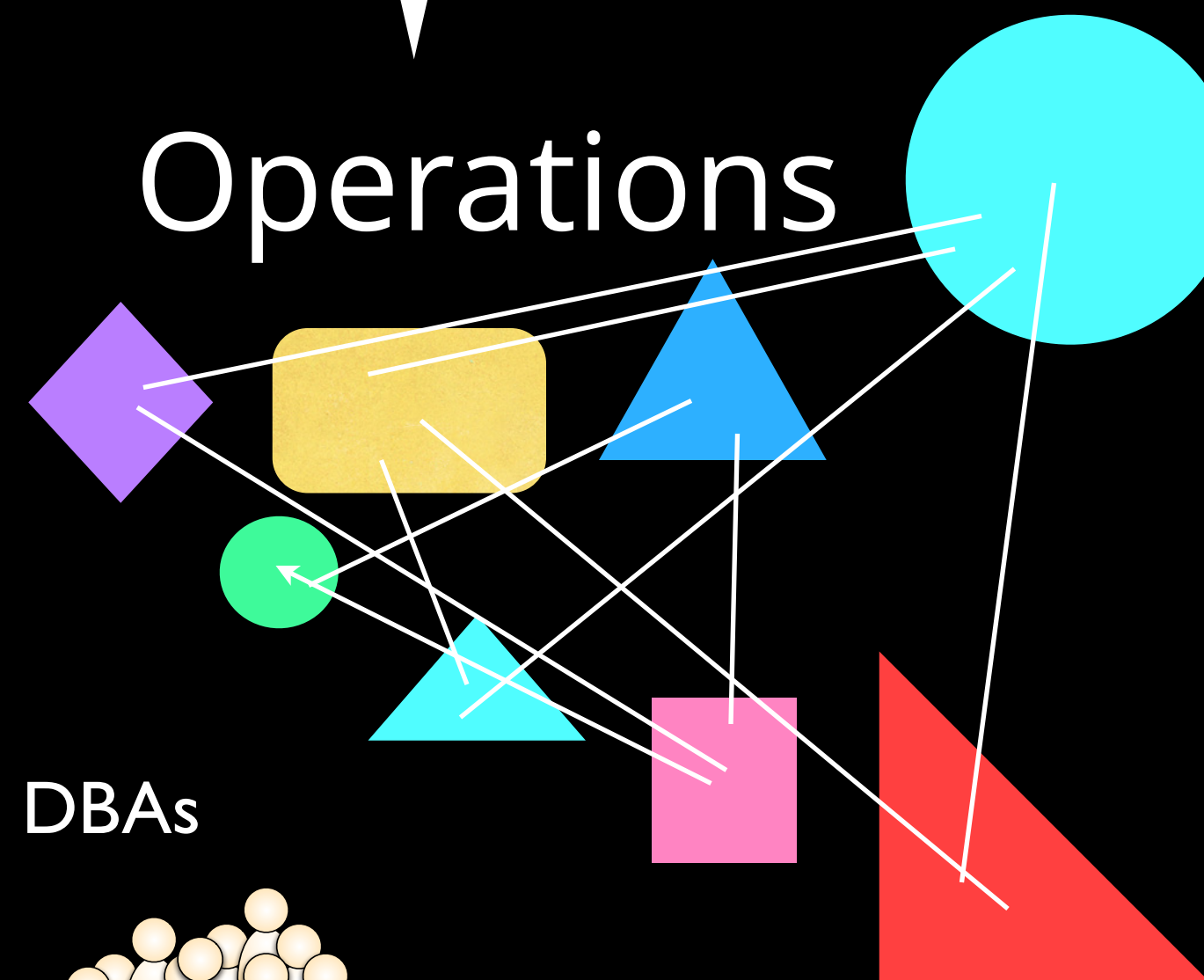
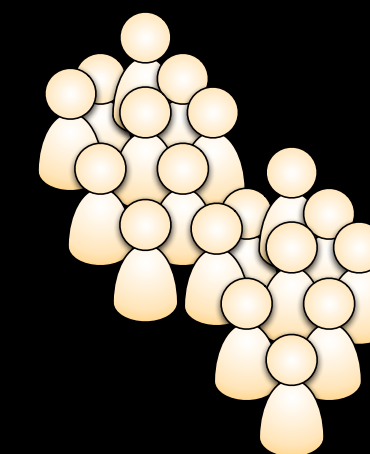
DBAs



Service desk



Infrastructure team



Value stream



# Amazon May Deployment Stats

(production hosts & environments only)

11.6 seconds

Mean time between deployments (weekday)

1,079

Max # of deployments in a single hour

10,000

Mean # of hosts simultaneously receiving a deployment

30,000

Max # of hosts simultaneously receiving a deployment

# a note about predictive analysis

---

One of three conditions must be met:

- Randomized, experimental design (no, this is non-experimental)
- Longitudinal (no, this is cross-sectional)
- Theory-based design

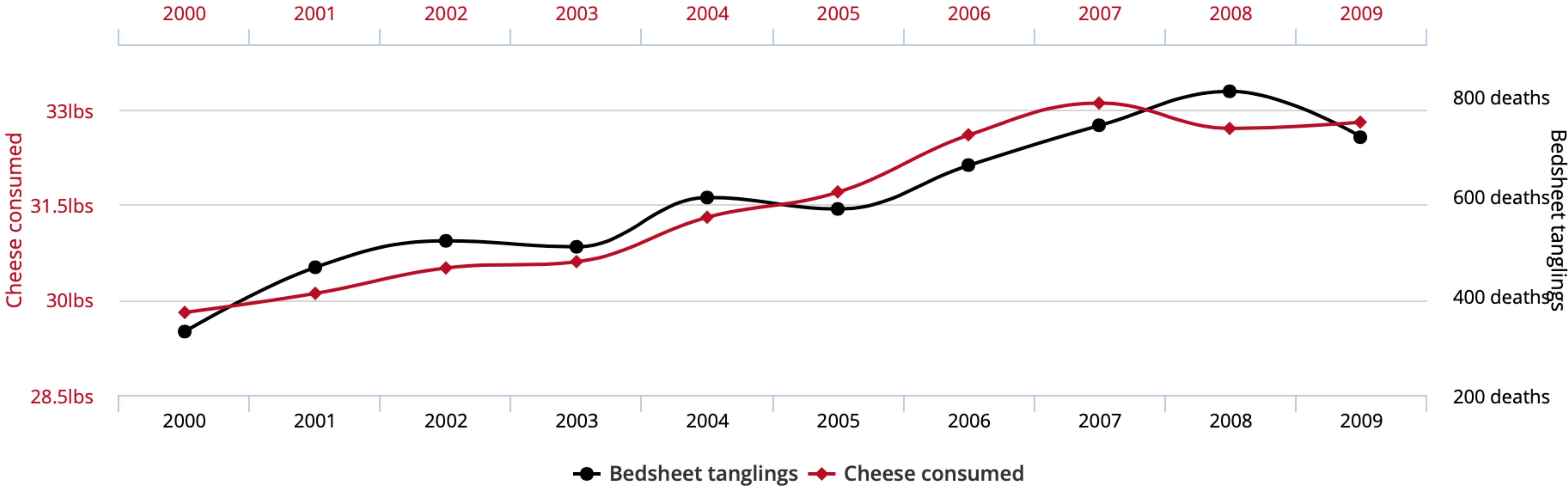
When this condition was not met, only correlations were tested and reported .

# Per capita cheese consumption

correlates with

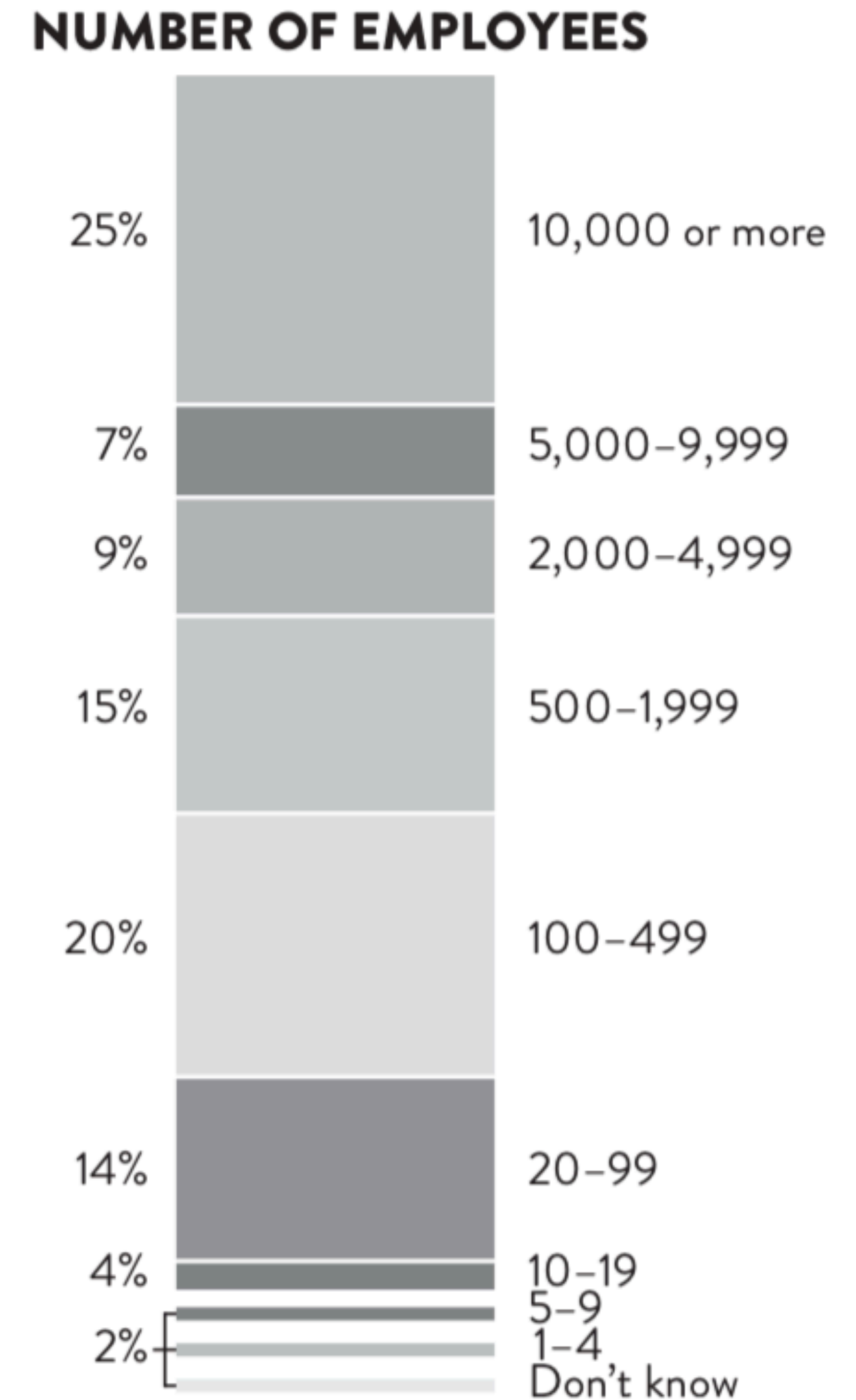
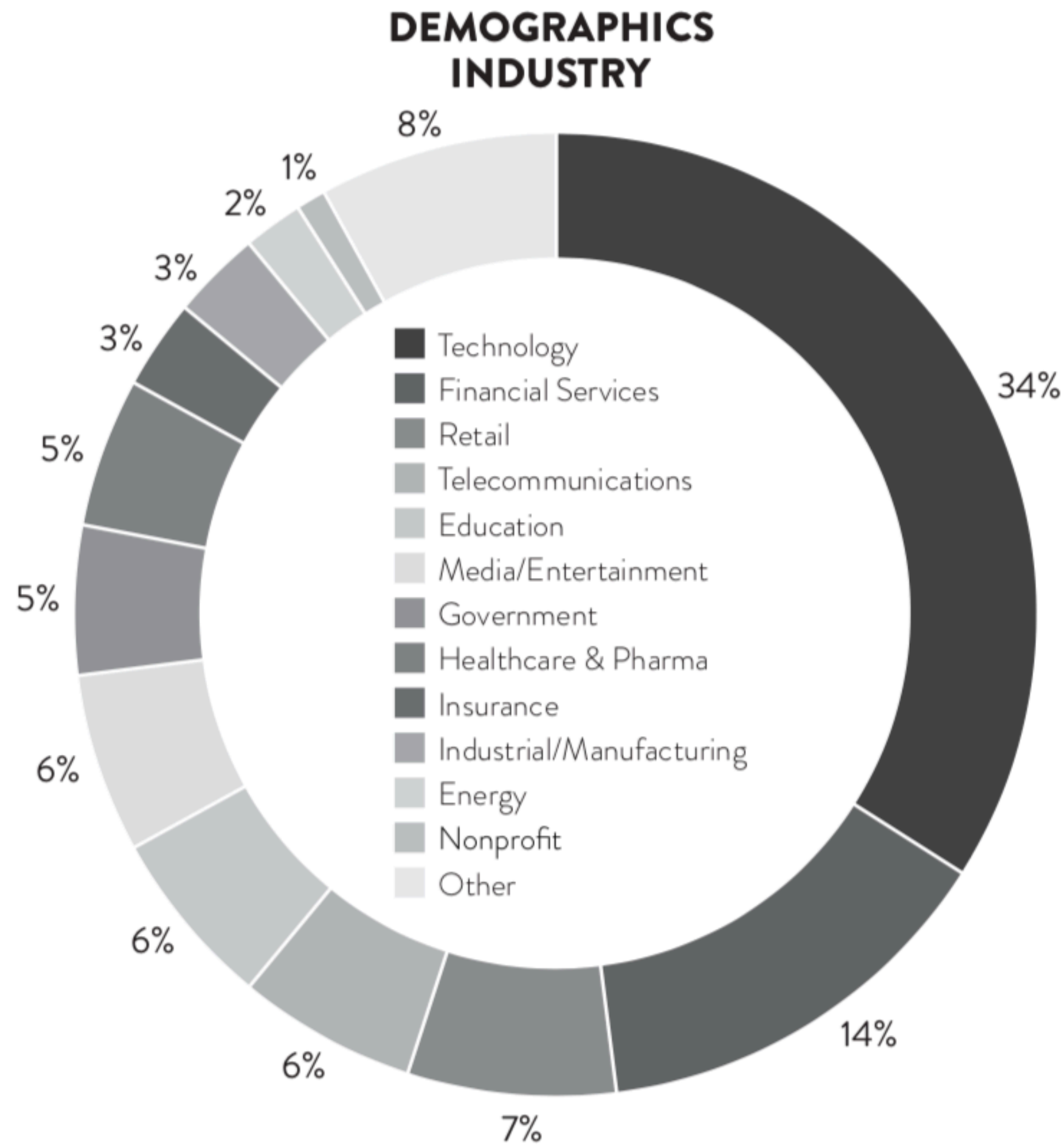
## Number of people who died by becoming tangled in their bedsheets

Correlation: 94.71% (r=0.947091)



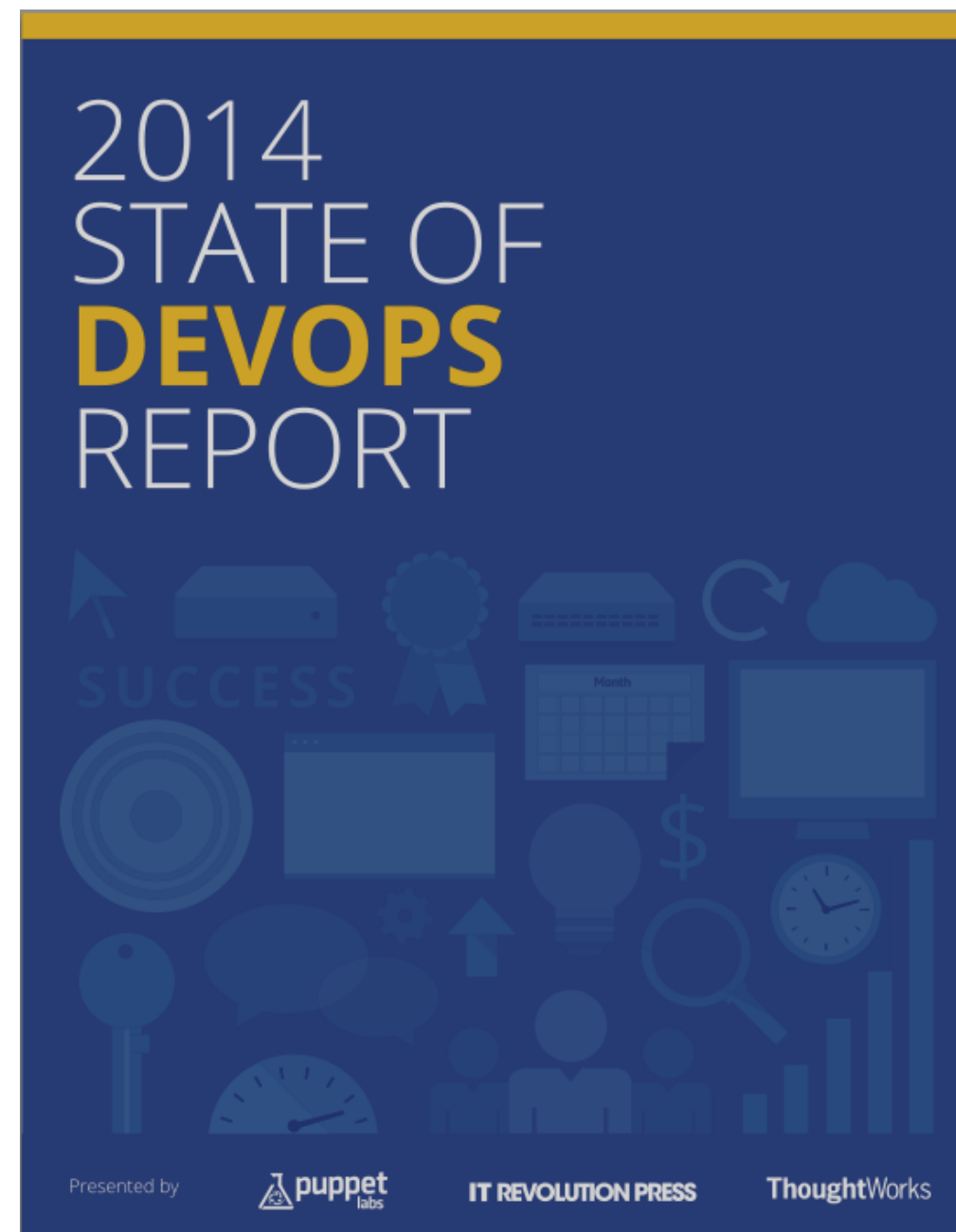


# firmographics



# software delivery as a competitive advantage

---



“Firms with high-performing IT organizations were *twice as likely* to exceed their *profitability, market share* and *productivity* goals.”

<http://bit.ly/2014-devops-report>

# software delivery as a competitive advantage



<http://bit.ly/2018-devops-report>

high performers were more than twice as likely to achieve or exceed the following objectives:

- Quantity of products or services
- Operating efficiency
- Customer satisfaction
- Quality of products or services provided
- Achieving organizational and mission goals
- Measures that demonstrate to external parties whether or not the organization is achieving intended results

# software delivery performance

---

lead time for changes (version control to production)

deploy frequency

time to restore service

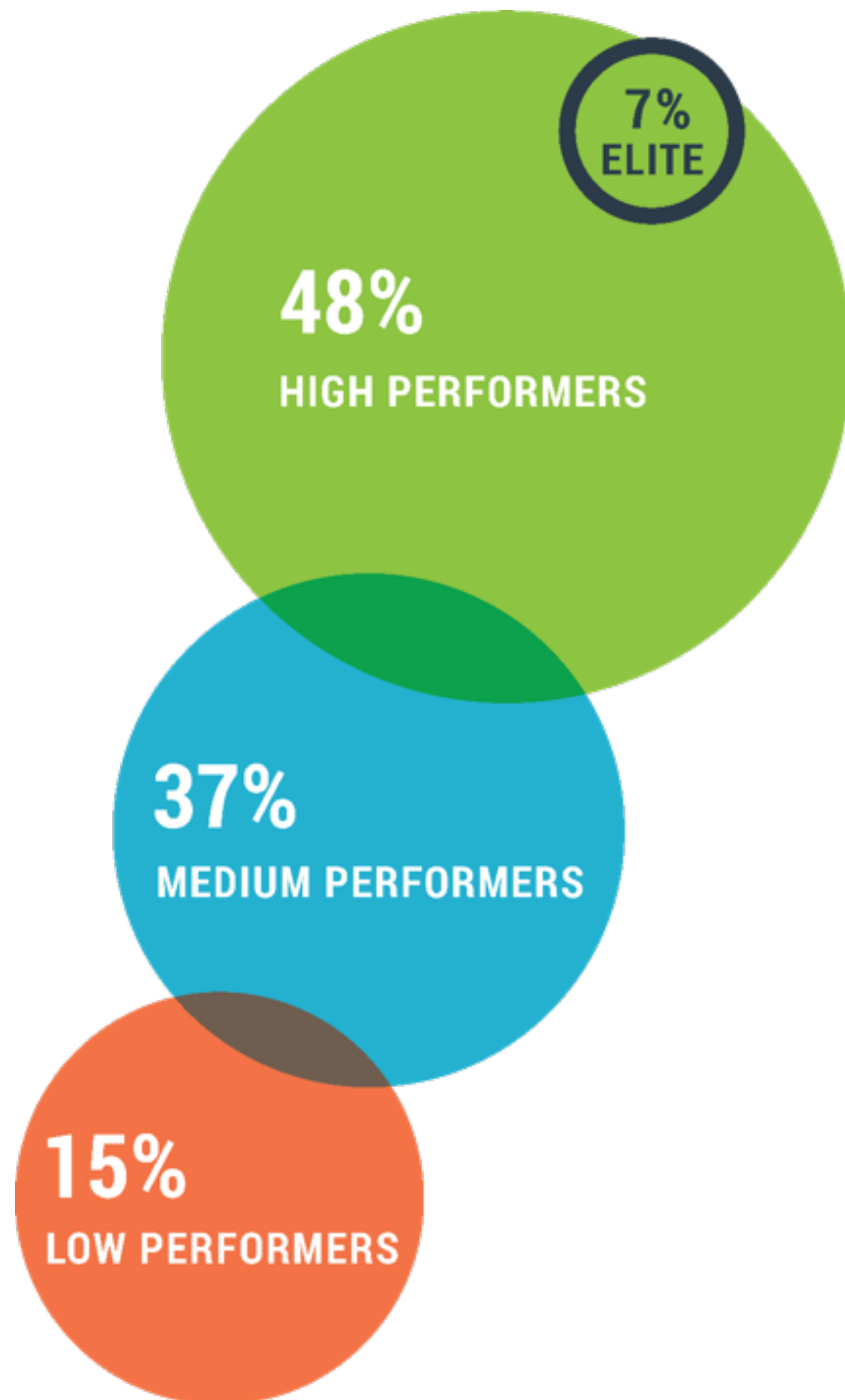
change fail rate

<http://bit.ly/2014-devops-report>

# 2018 performance benchmarks

Aspect of Software Delivery Performance	Elite <sup>a</sup>	High	Medium	Low
<b>Deployment frequency</b> For the primary application or service you work on, how often does your organization deploy code?	On-demand (multiple deploys per day)	Between once per hour and once per day	Between once per week and once per month	Between once per week and once per month
<b>Lead time for changes</b> For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code commit to code successfully running in production)?	Less than one hour	Between one day and one week	Between one week and one month <sup>b</sup>	Between one month and six months <sup>b</sup>
<b>Time to restore service</b> For the primary application or service you work on, how long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment)?	Less than one hour	Less than one day	Less than one day	Between one week and one month
<b>Change failure rate</b> For the primary application or service you work on, what percentage of changes results either in degraded service or subsequently requires remediation (e.g., leads to service impairment, service outage, requires a hotfix, rollback, fix forward, patch)?	0-15%	0-15%	0-15%	46-60%

# elite performers



Data shows a new 4th high performance group:  
**elite performers**

Proportion of high performers has grown YoY,  
but the bar for excellence remains high

Elite performers are still able to optimize for  
throughput and stability

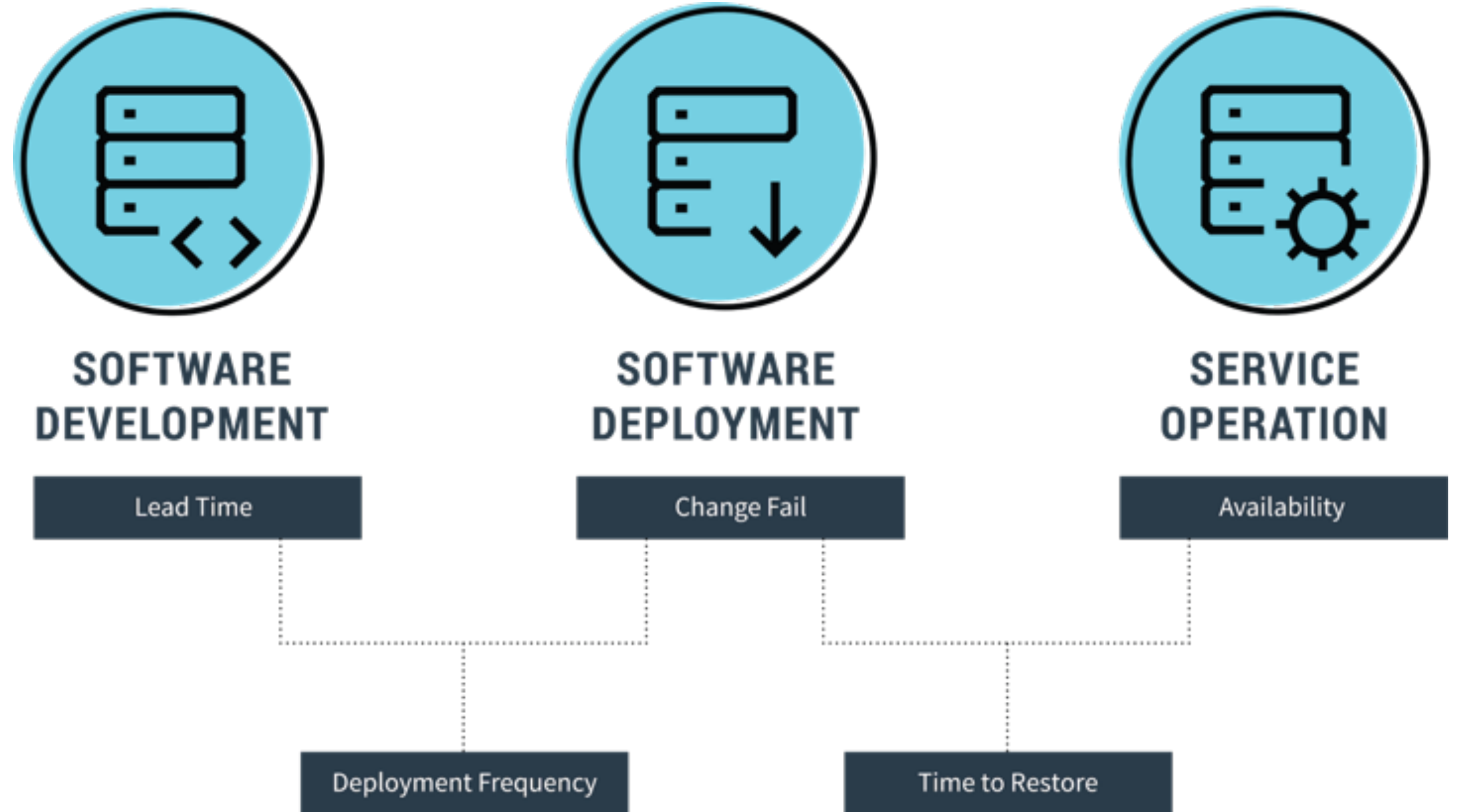
<http://bit.ly/2018-devops-report>

# availability

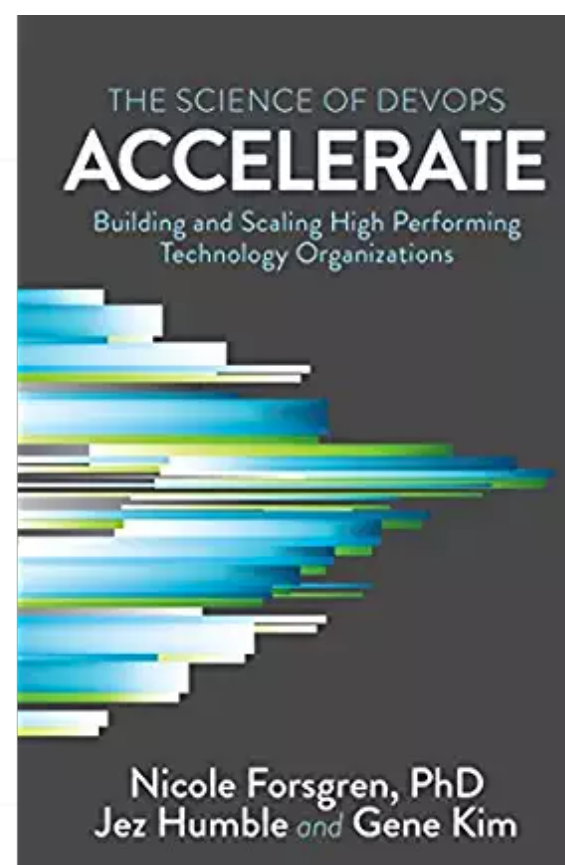
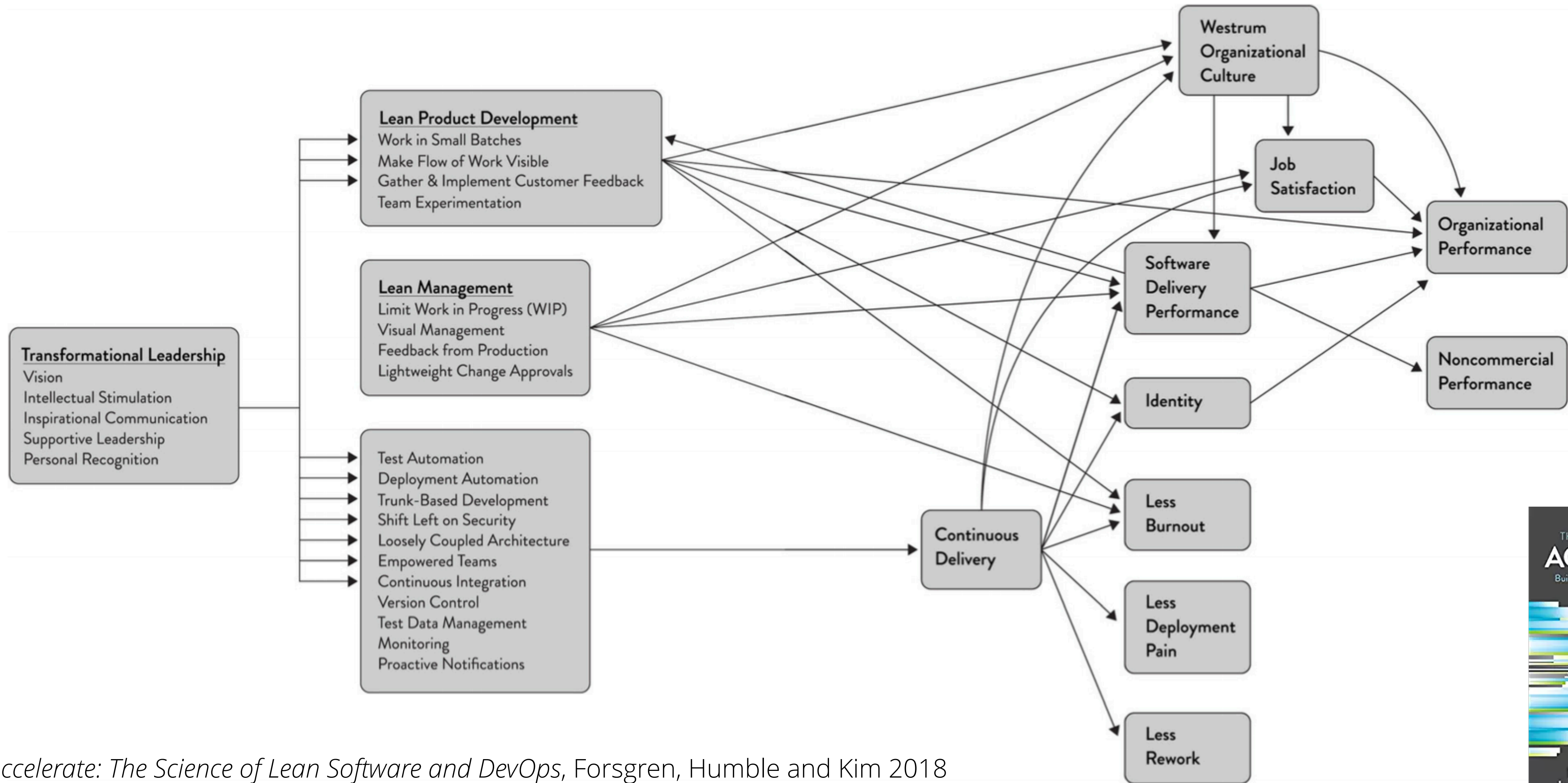
Ability for teams to ensure their product or service can be accessed by end users

Software delivery + availability = **SDO performance**

Elite performers are **3.5X more likely** to have strong availability practices

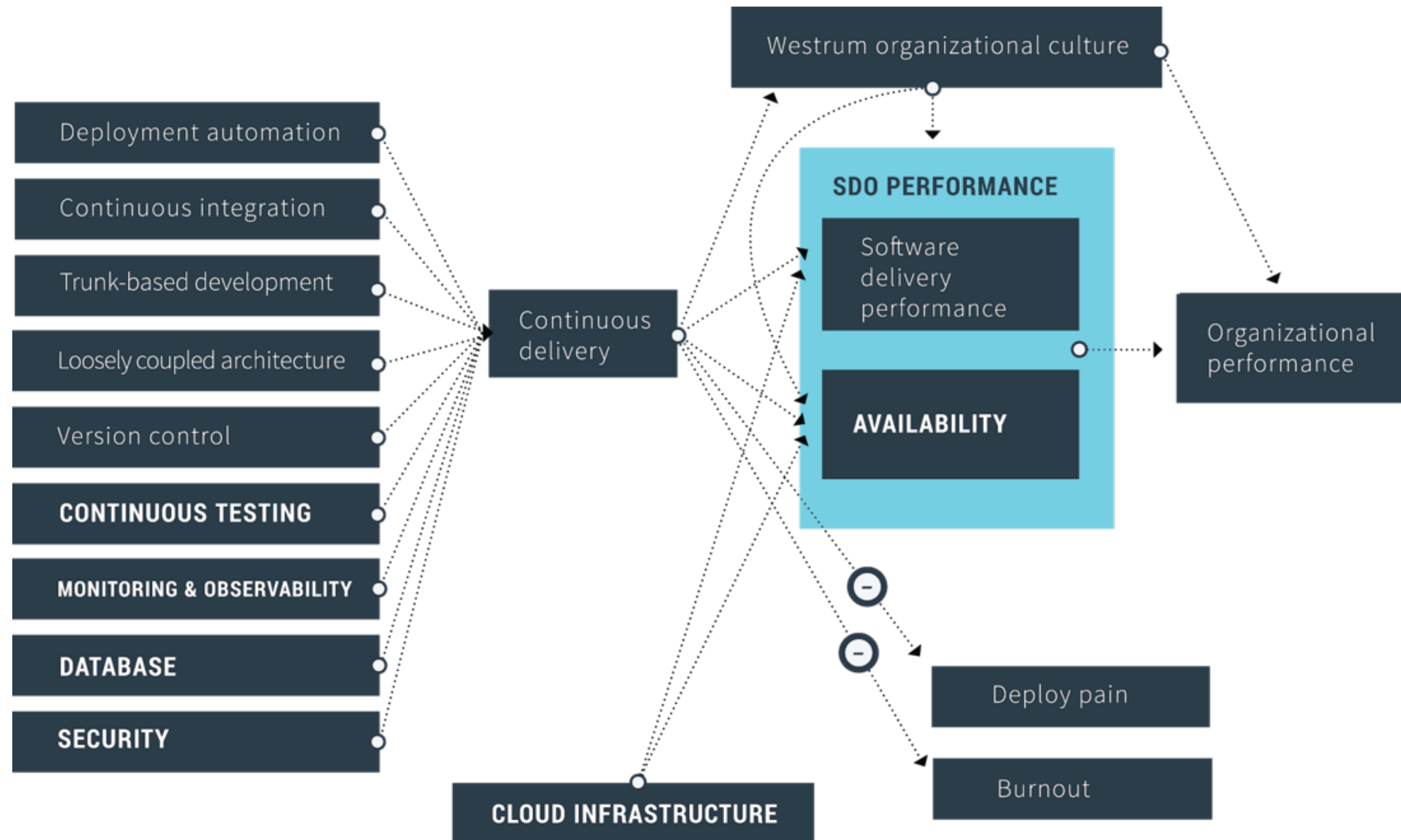


# capabilities that drive high performance





# technical practices



# key finding: architectural outcomes

---

## **can my team...**

...make large-scale changes to the design of its system without the permission of somebody outside the team or depending on other teams?

...complete its work without needing fine-grained communication and coordination with people outside the team?

...deploy and release its product or service on demand, independently of other services the product or service depends upon?

...do most of its testing on demand, without requiring an integrated test environment?

...perform deployments during normal business hours with negligible downtime?

# key finding: doing cloud right

%

AGREED OR STRONGLY AGREED

46%

**On-demand self-service**

Only 22% of teams are doing cloud right!

46%

**Broad network access**

Teams that use these essentials characteristics are **23X more likely** to be elite performers

43%

**Resource Pooling**

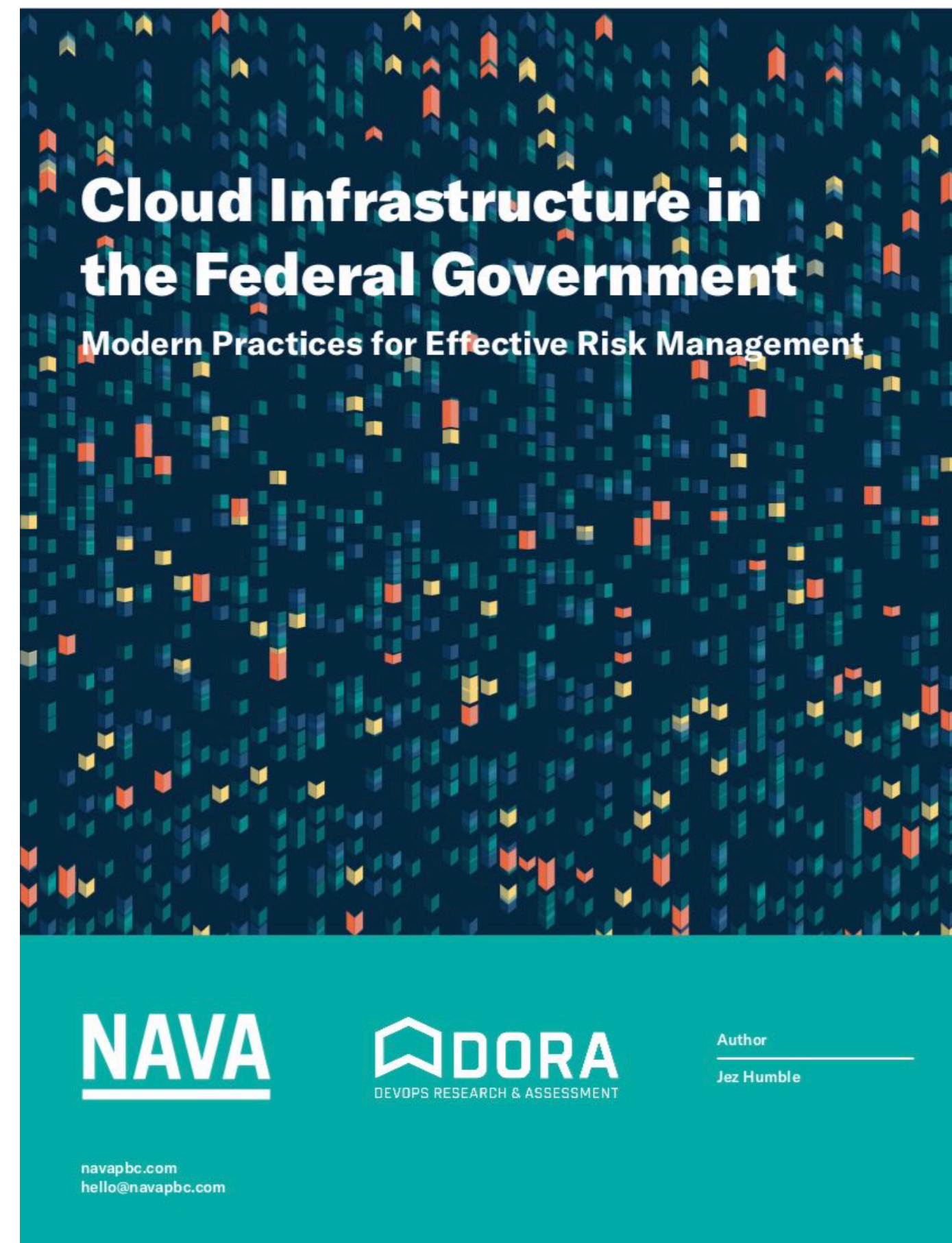
45%

**Rapid elasticity**

48%

**Measured service**

# cloud in regulated environments



<https://devops-research.com/research.html>

@jezhumble

# monitoring and observability

Teams with a comprehensive monitoring and observability solution were **1.3 times more likely to be an elite performer.**

Having a monitoring and observability solution **positively contributed to SDO performance.**

Fun stats fact: monitoring and observability load together.

## MONITORING

is tooling or a technical solution that allows teams to watch and understand the state of their systems and is based on gathering predefined sets of metrics or logs.

## OBSERVABILITY

is tooling or a technical solution that allows teams to actively debug their system and explore properties and patterns they have not defined in advance.

# which of these measure effective test practices?

---

- Developers primarily create & maintain acceptance tests
- QA primarily create & maintain acceptance tests
- Primarily created & maintained by outsourced party
- When automated tests pass, I'm confident the software is releasable
- Test failures are likely to indicate a real defect
- It's easy for developers to fix acceptance tests
- Developers share a common pool of test servers to reproduce failures
- Developers create on demand test environments
- Developers use their own dev environments to reproduce failures

# which of these measure effective test practices?

---

- Developers primarily create & maintain acceptance tests
- QA primarily create & maintain acceptance tests
- Primarily created & maintained by outsourced party
- When automated tests pass, I'm confident the software is releasable
- Test failures are likely to indicate a real defect
- It's easy for developers to fix acceptance tests
- Developers share a common pool of test servers to reproduce failures
- Developers create on demand test environments
- Developers use their own dev environments to reproduce failures

# continuous testing

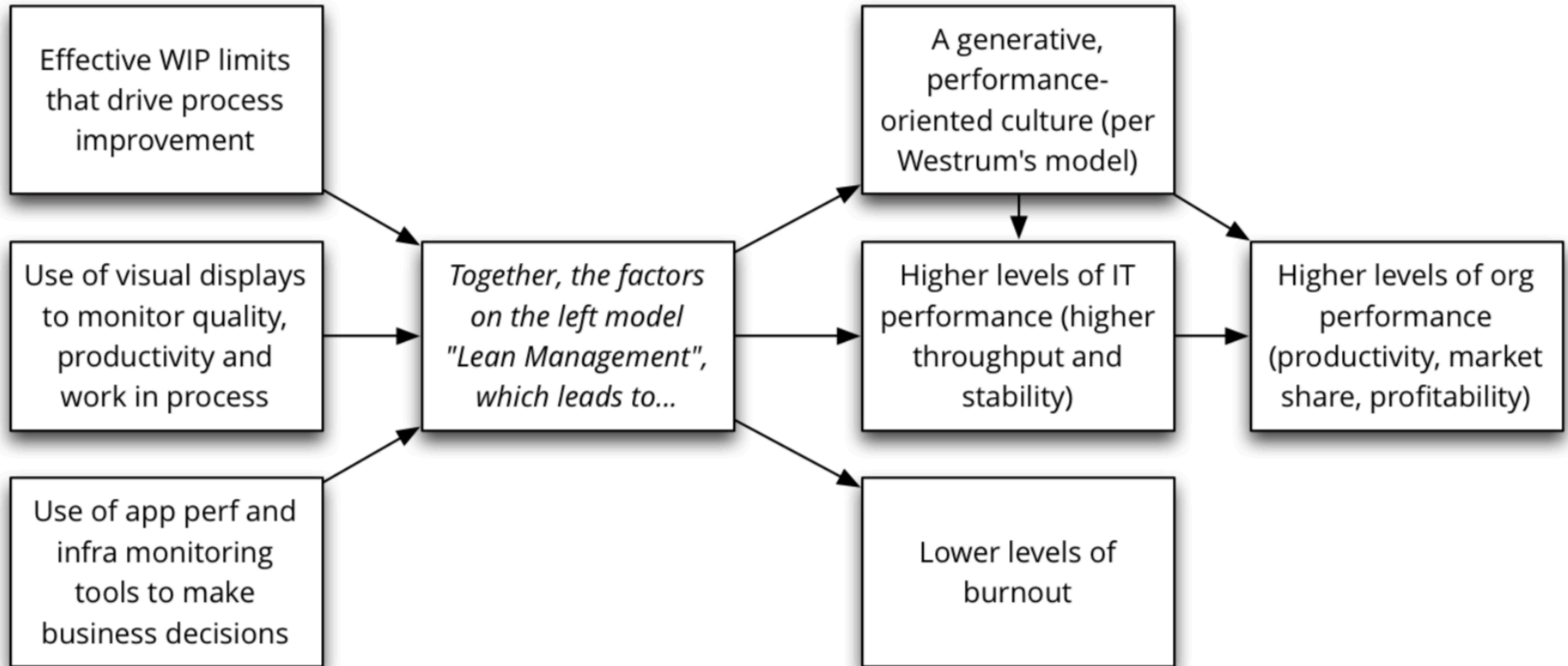
---

## **previous practices plus...**

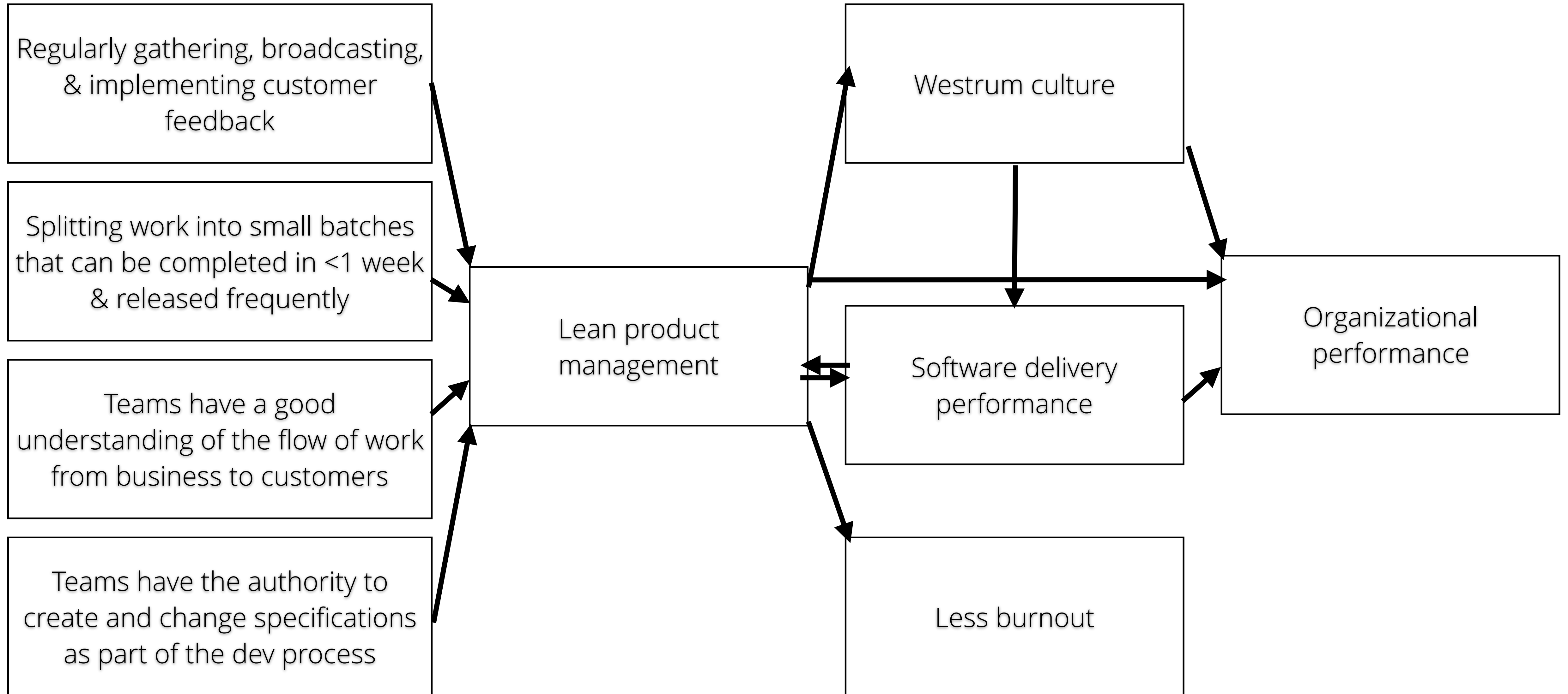
- continuously reviewing and improving test suites to better find defects and keep complexity and cost under control
- allowing testers to work alongside developers throughout the software development and delivery process
- performing manual test activities such as exploratory testing, usability testing, and acceptance testing throughout the delivery process
- having developers practice test-driven development by writing unit tests before writing production code for all changes to the codebase
- being able to get feedback from automated tests in less than ten minutes both on local workstations and from a CI server



# lean management

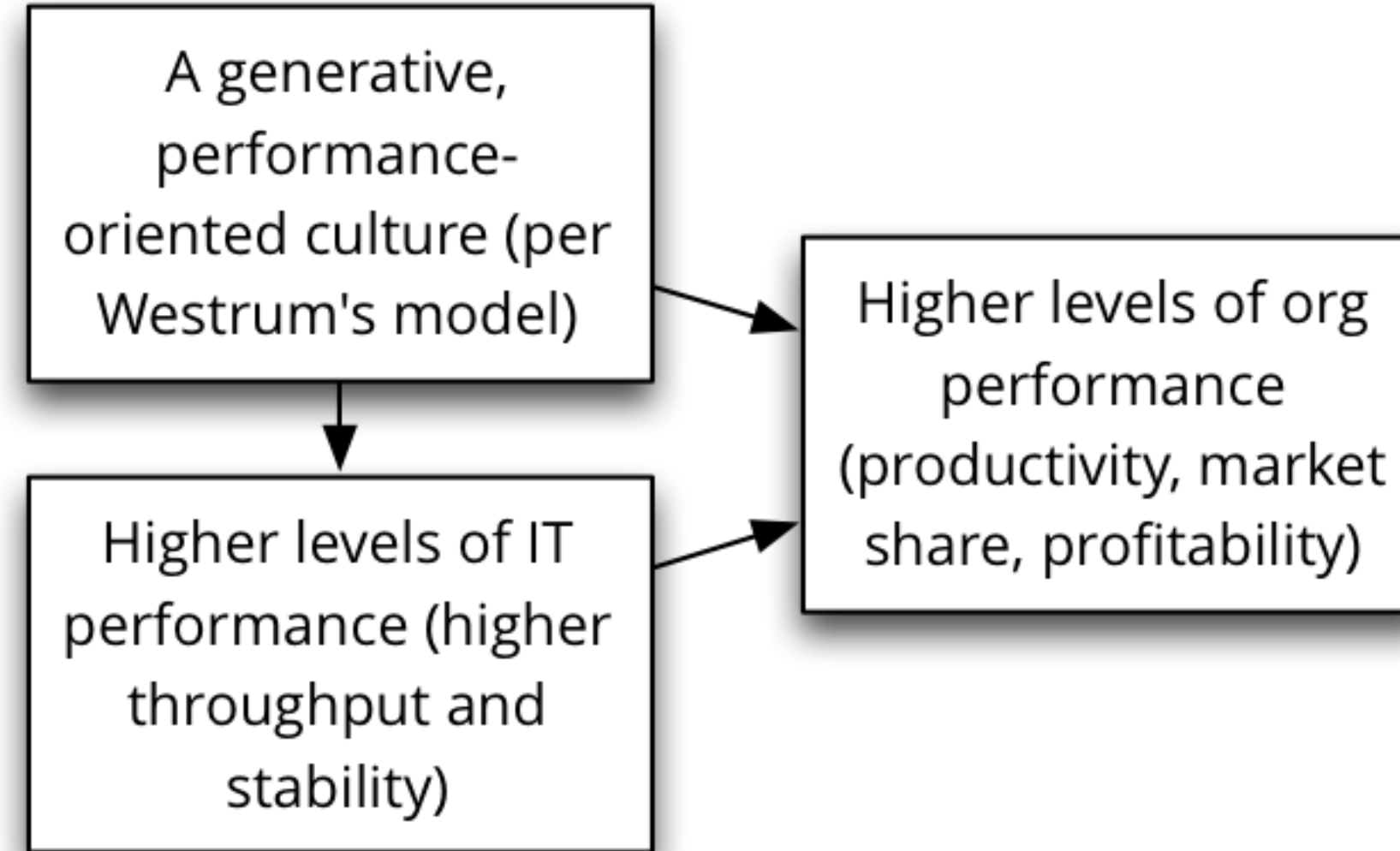


# lean product management



# culture impacts performance

---



# high trust culture

how organizations process information

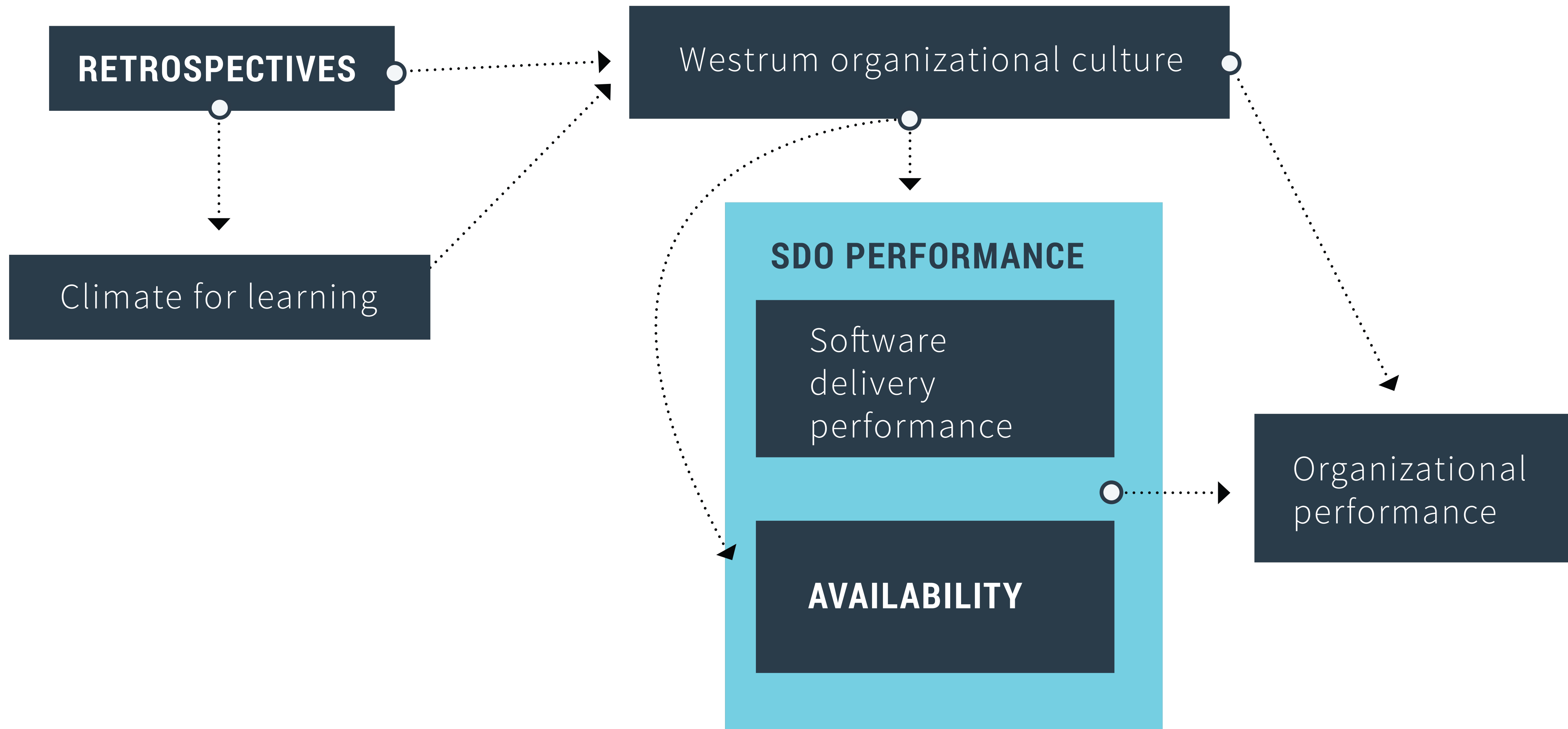
<b>Pathological (<i>power oriented</i>)</b>	<b>Bureaucratic (<i>rule oriented</i>)</b>	<b>Generative (<i>performance oriented</i>)</b>
Low cooperation	Modest cooperation	High cooperation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to enquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

# effective teams

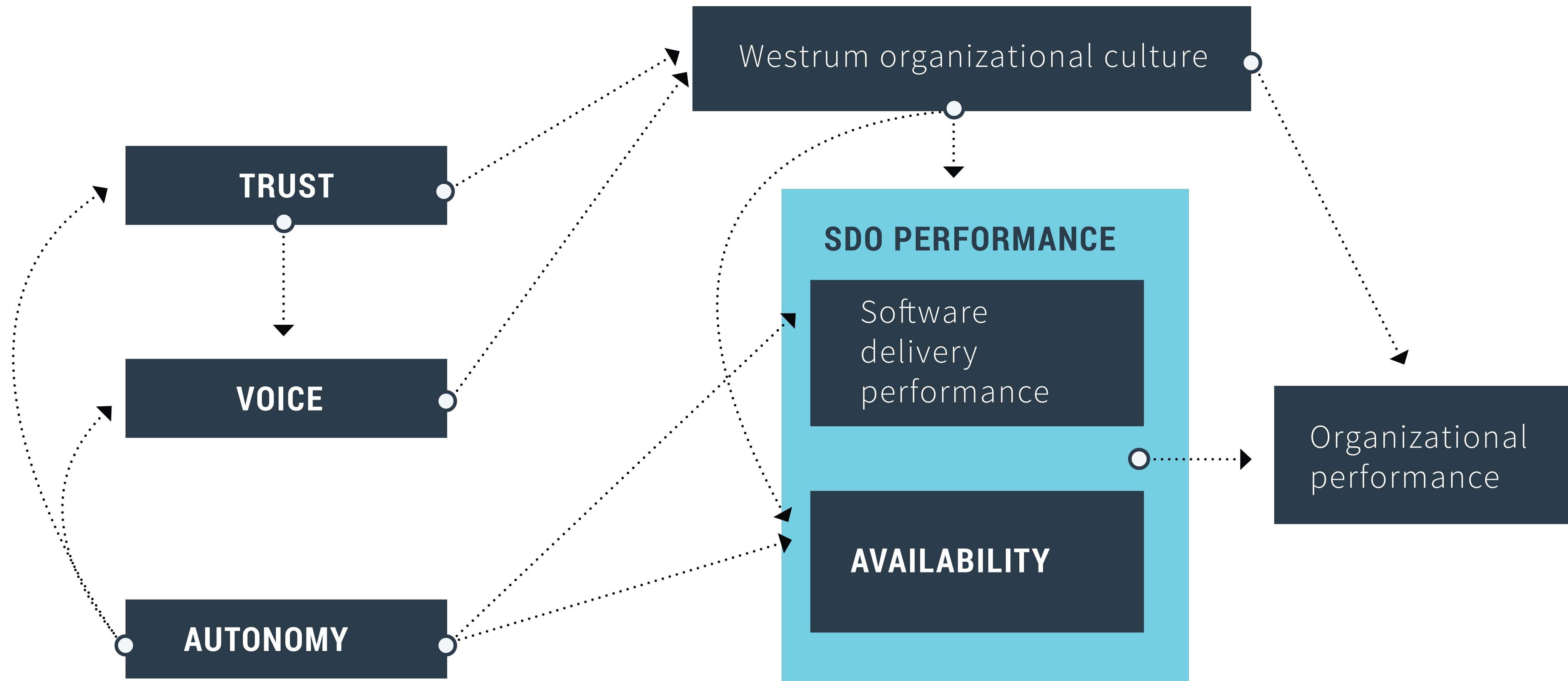


<https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/>

# climate for learning



# autonomy



# Highly Aligned, Loosely Coupled

- Highly Aligned
  - Strategy and goals are clear, specific, broadly understood
  - Team interactions focused on strategy and goals, rather than tactics
  - Requires large investment in management time to be transparent and articulate and perceptive
- Loosely Coupled
  - Minimal cross-functional meetings except to get aligned on goals and strategy
  - Trust between groups on tactics without previewing/approving each one – so groups can move fast
  - Leaders reaching out proactively for ad-hoc coordination and perspective as appropriate
  - Occasional post-mortems on tactics necessary to increase alignment



# innovation culture

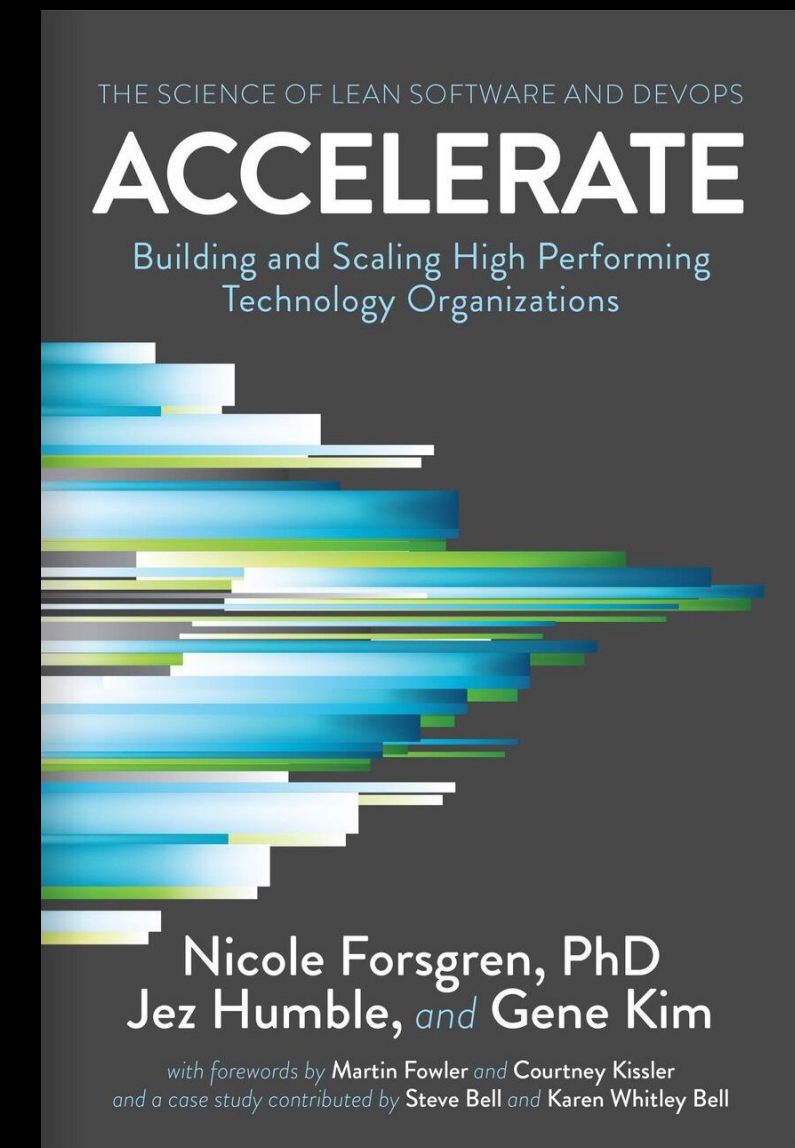
---

“I think building this culture is the key to innovation. Creativity must flow from everywhere. Whether you are a summer intern or the CTO, any good idea must be able to seek an objective test, preferably a test that exposes the idea to real customers. Everyone must be able to experiment, learn, and iterate.”

# thank you!

To receive the following:

- An excerpt from *Accelerate*
- 30% off Jez's new video course: creating high performance organizations
- 50% off Jez's CD video training, interviews with Eric Ries, and more
- A copy of this presentation
- A 100 page excerpt from *Lean Enterprise*
- An excerpt from *The DevOps Handbook*
- A 20m preview of Jez's Continuous Delivery video workshop



Just pick up your phone and send an email

To: [jezhumble@sendyourslides.com](mailto:jezhumble@sendyourslides.com)

Subject: devops



© 2016-7 DevOps Research and Assessment LLC  
<https://devops-research.com/>

