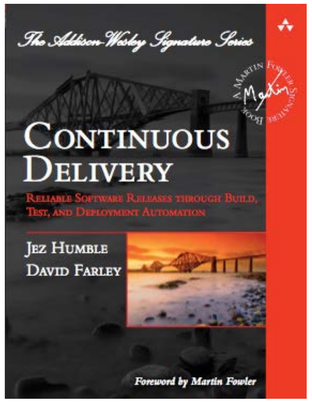
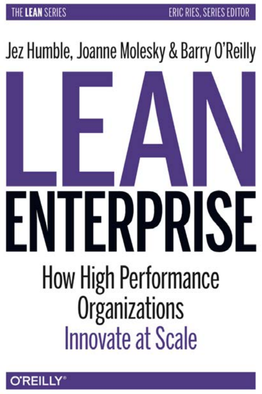
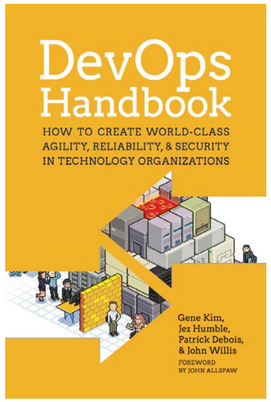
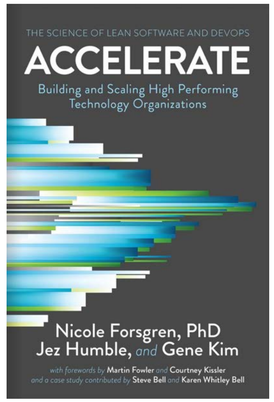
The key to high performing tech organizations

@jezhumble #agileaus

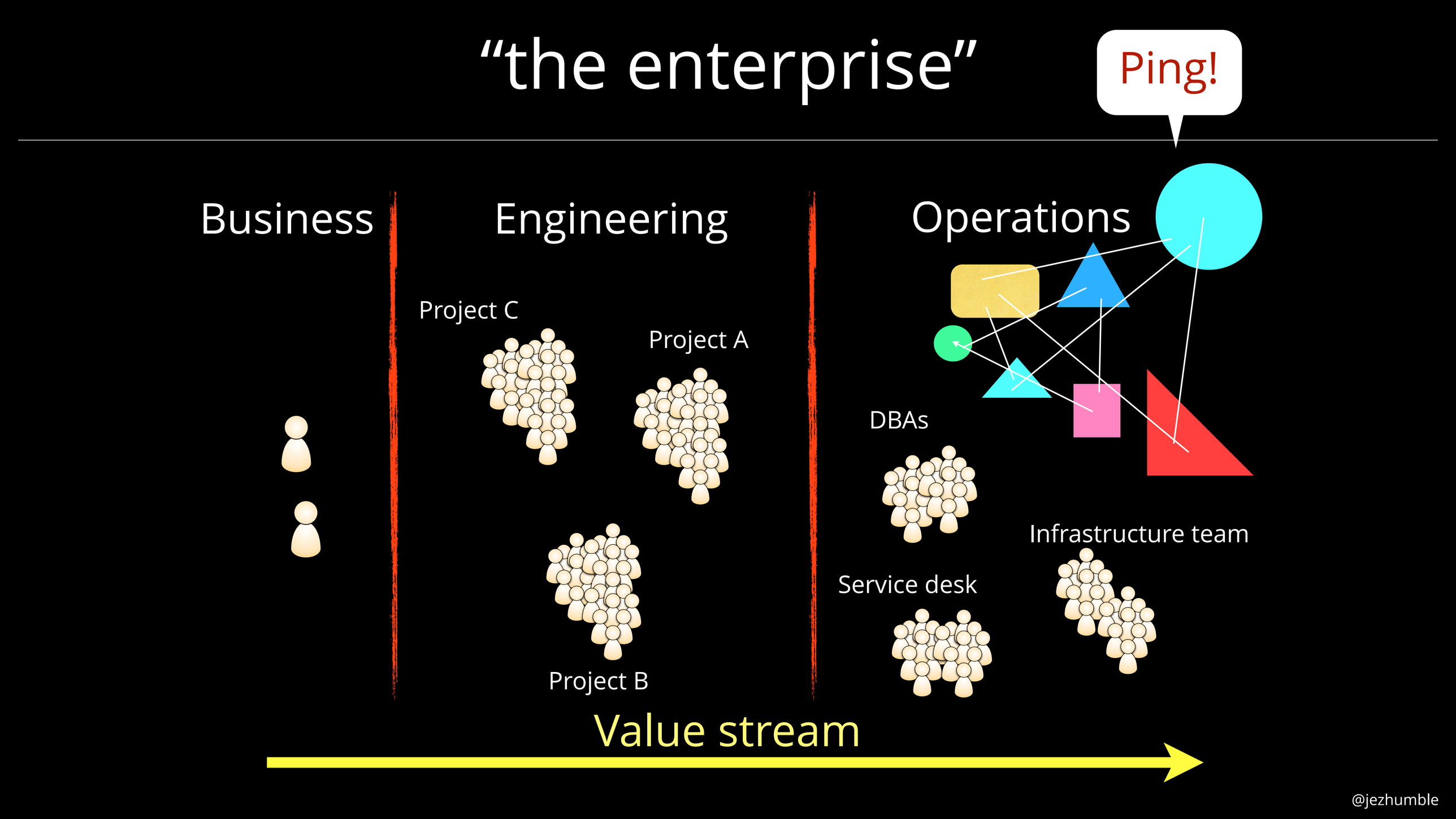




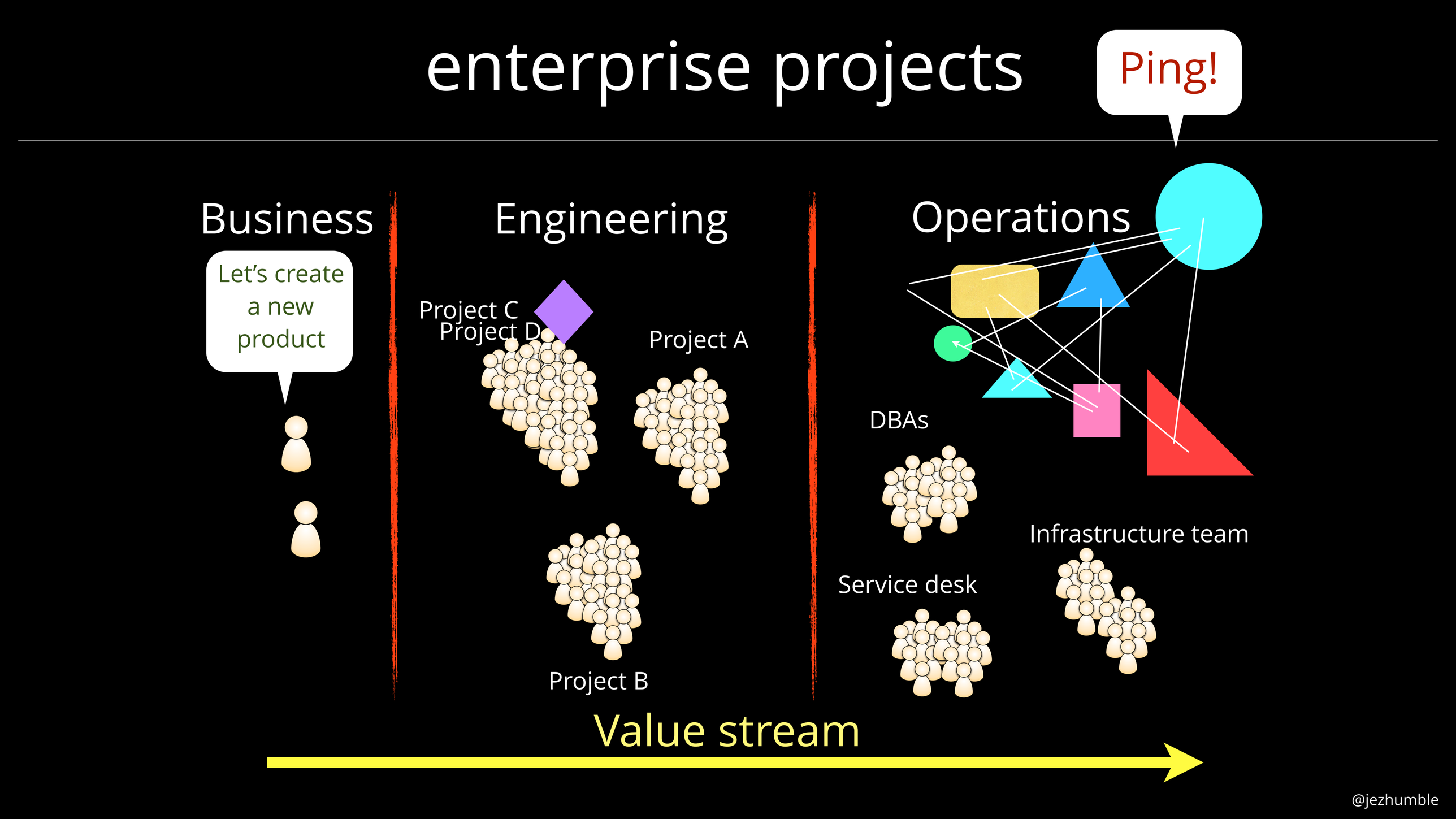




“the enterprise”



enterprise projects



Oh no!

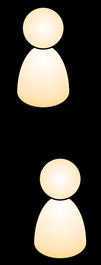
We're going agile!

Oh no!

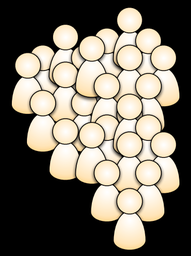
Business

Engineering

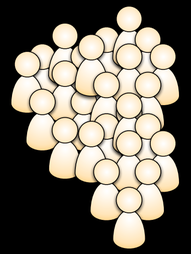
Operations

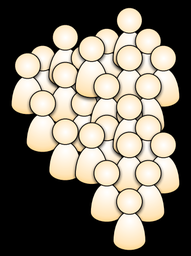


Project D

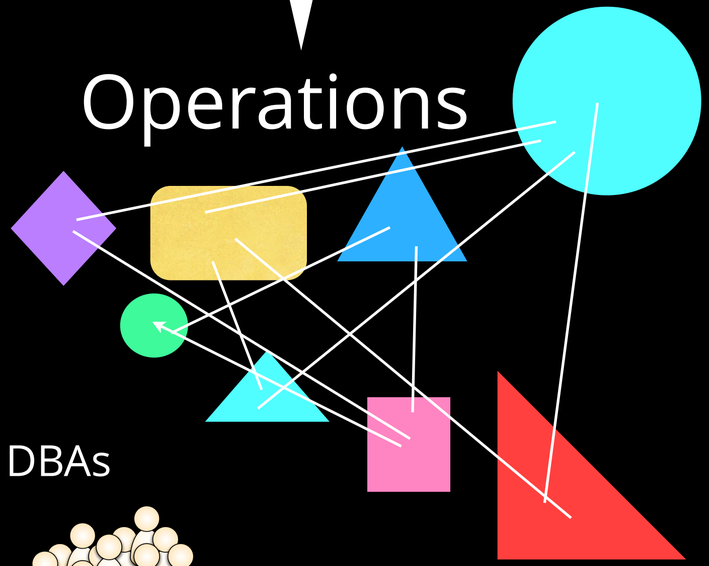


Project A

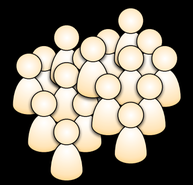




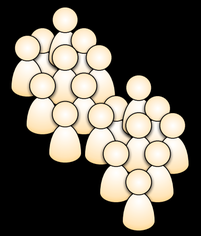
Project B



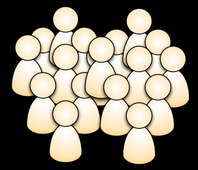
DBAs



Infrastructure team



Service desk



Value stream

Our test-driven code follows SOLID principles

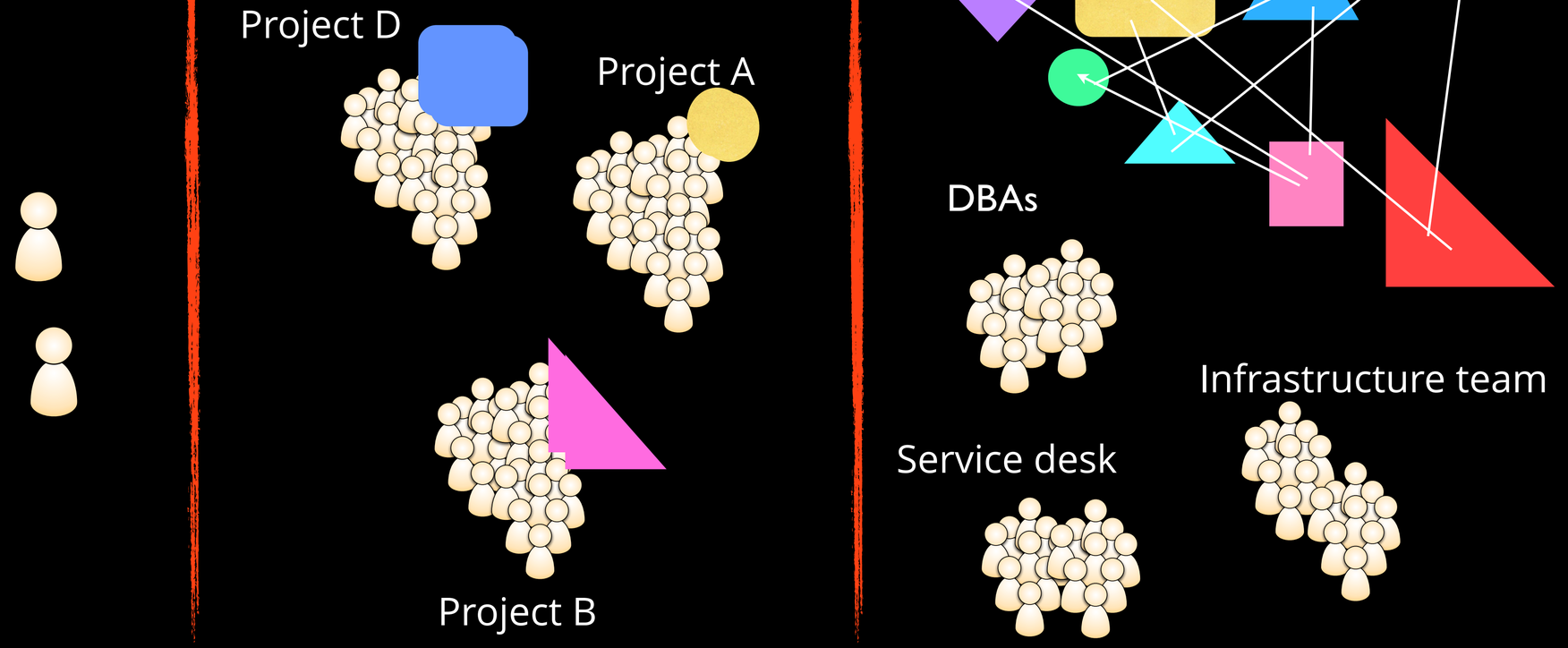
Change management

Shame it doesn't work

Business

Engineering

Operations



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

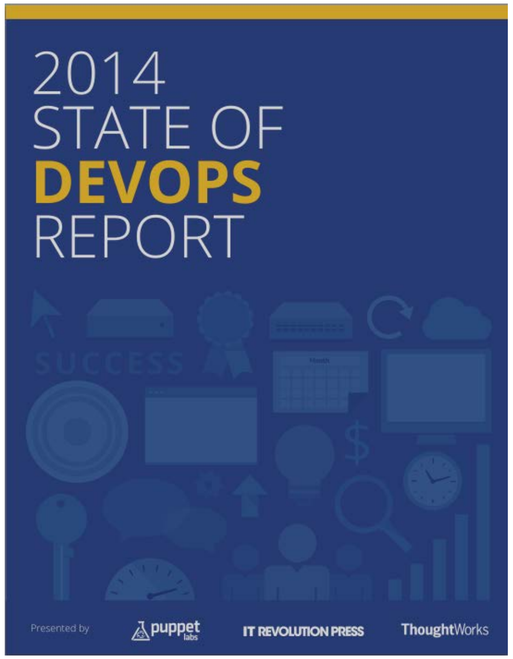
Mean # of hosts simultaneously receiving a deployment

30,000

Max # of hosts simultaneously receiving a deployment

Jon Jenkins, “Velocity Culture, The Unmet Challenge in Ops” | http://bit.ly/1vJo1Ya

software delivery as a competitive advantage



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

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

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

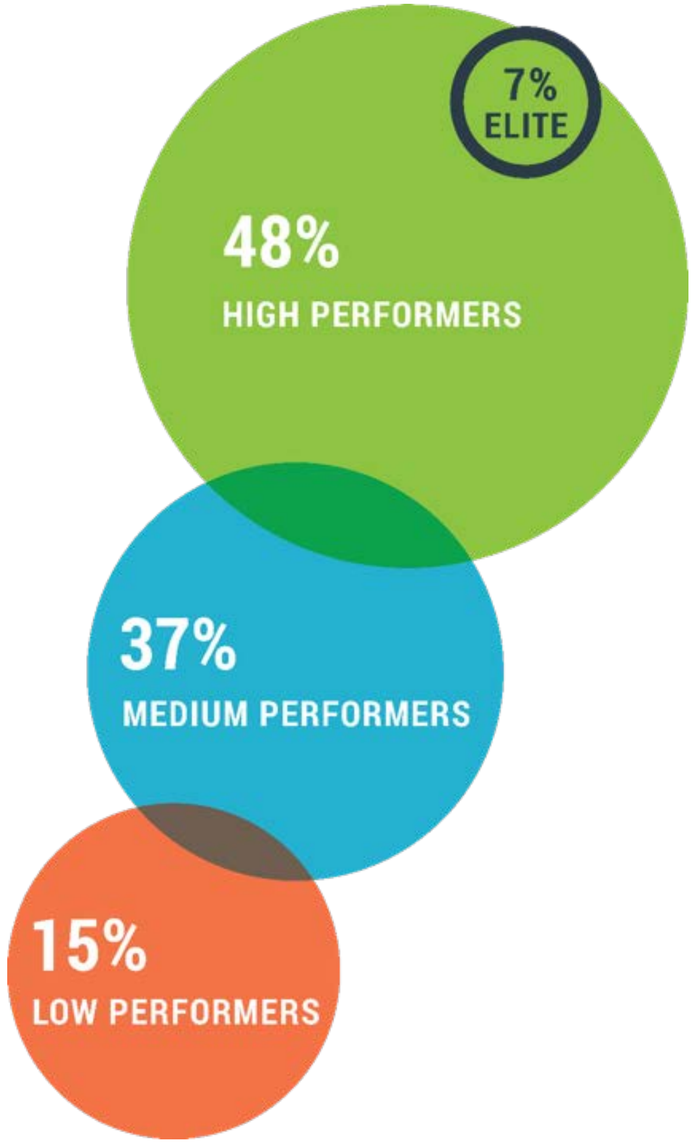
http://bit.ly/2014-devops-report

2018 performance benchmarks

Aspect of Software Delivery Performance	Elite[a]	High	Medium	Low
Deployment frequency 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
Lead time for changes 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[b]	Between one month and six months[b]
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 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
Change failure rate 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%

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

elite performers



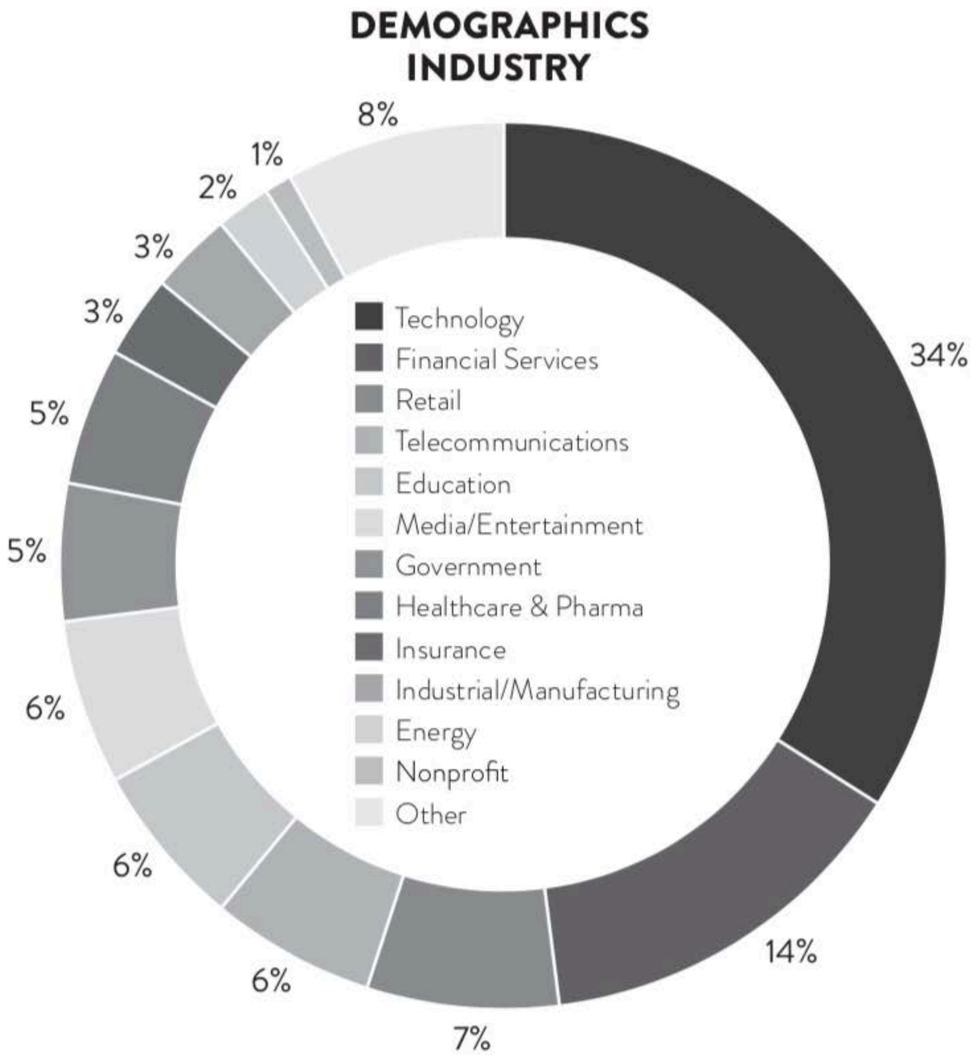
Data shows a new 4th high performance group:
elite performers

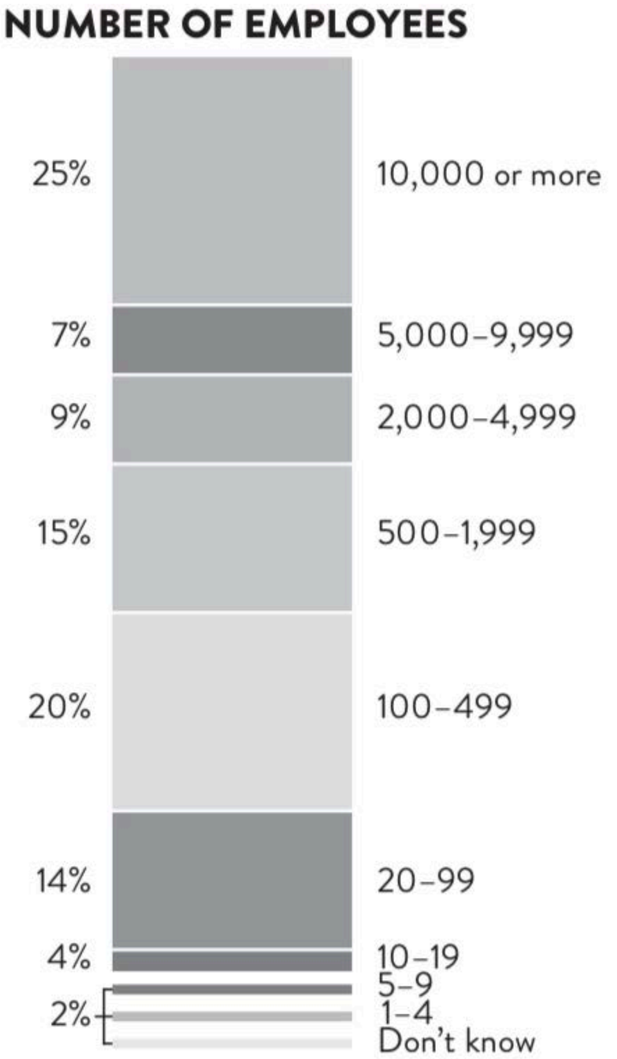
Proportion of high performers has grown YoY, but the bar for excellence remains high

Elite performers are still able to optimize for throughput and stability

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

firmographics





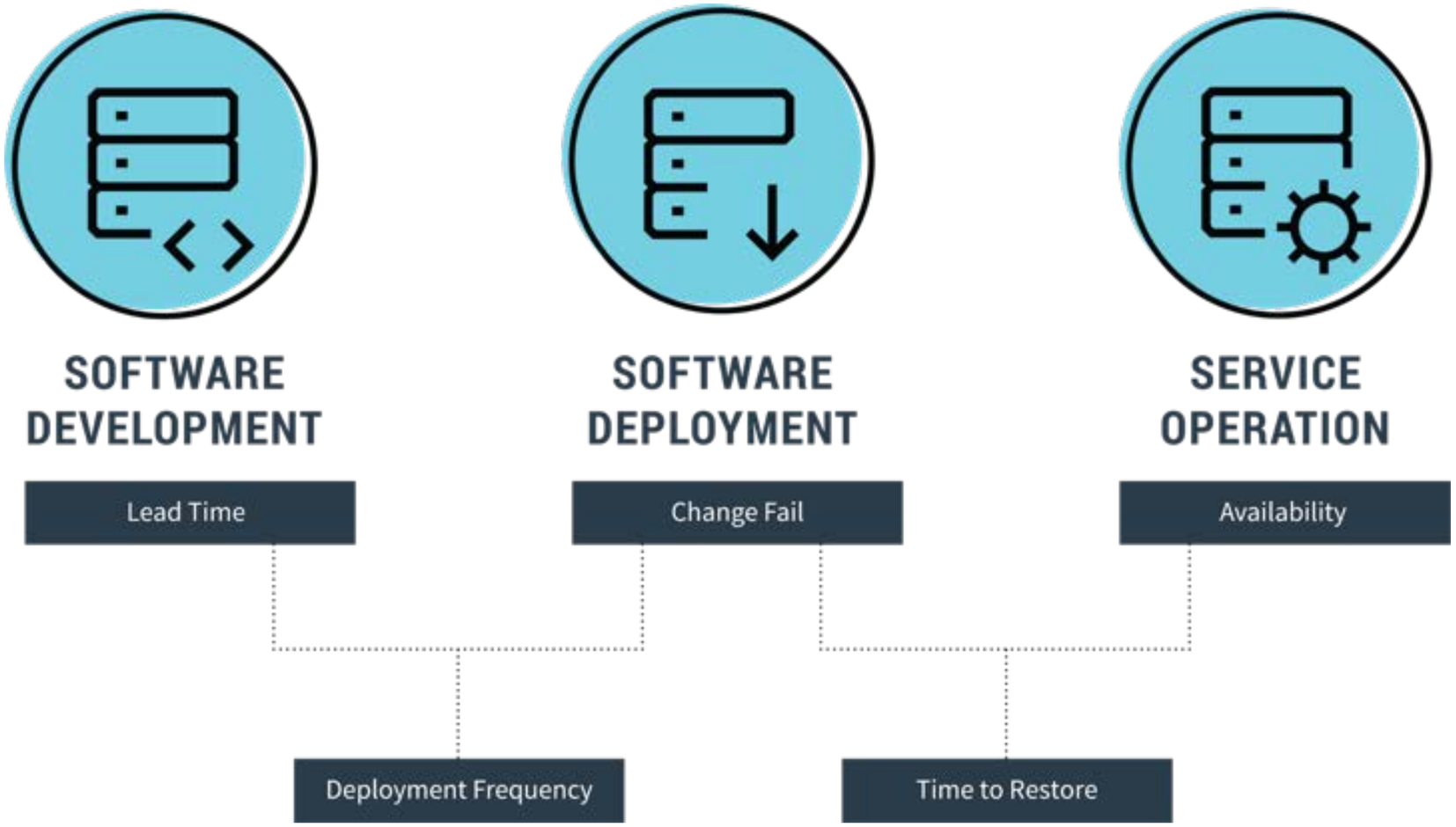
Accelerate: The Science of Lean Software and DevOps, Forsgren, Humble and Kim 2018

availability

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

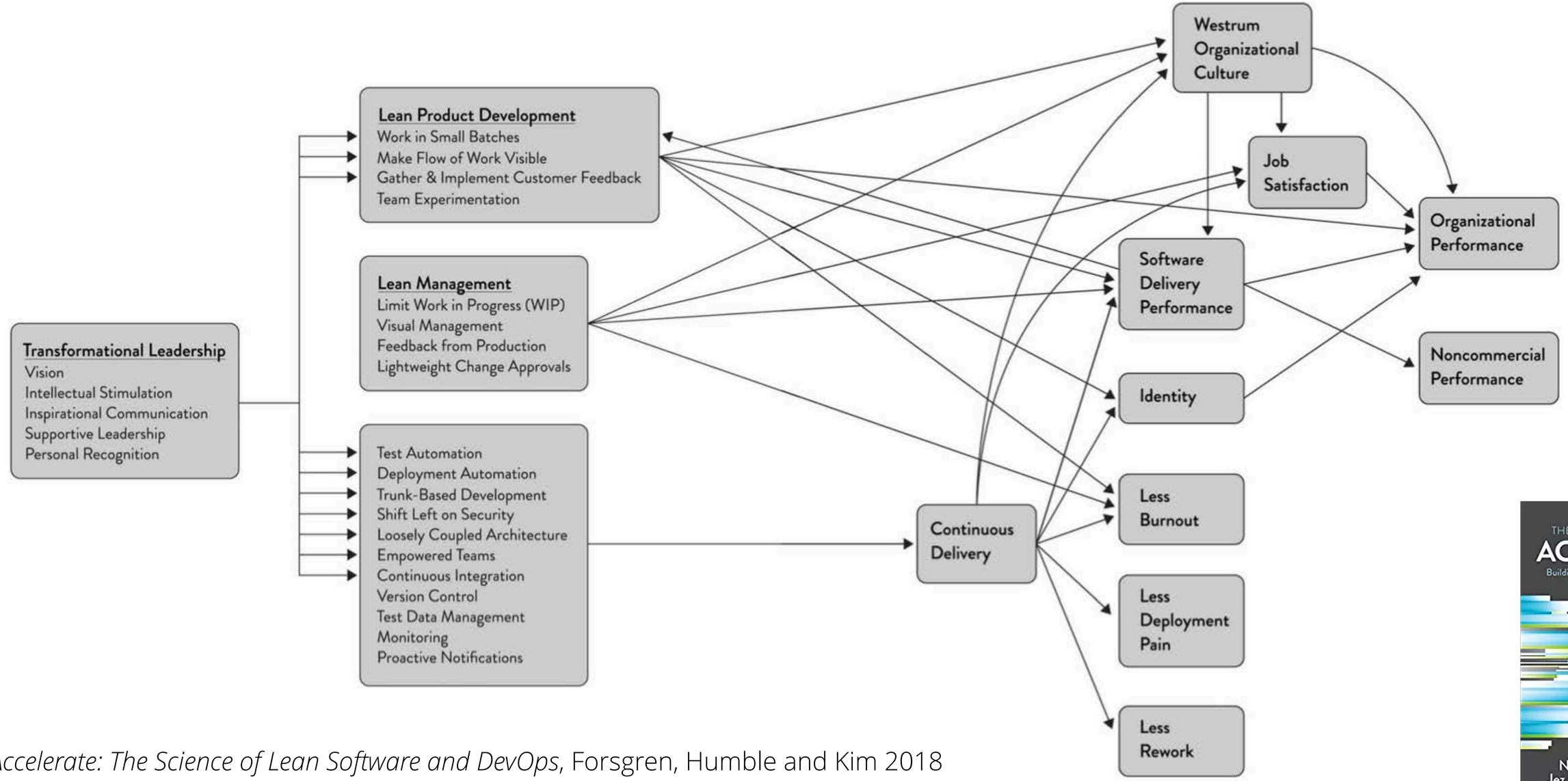
Software delivery + availability = **SDO performance**

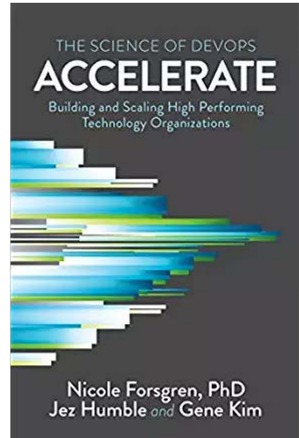
Elite performers are **3.5X more likely** to have strong availability practices



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

capabilities that drive high performance





Accelerate: The Science of Lean Software and DevOps, Forsgren, Humble and Kim 2018

what is continuous delivery?

The ability to get changes—features, configuration changes, bug fixes, experiments—into production or into the hands of users *safely* and *quickly* in a *sustainable* way.

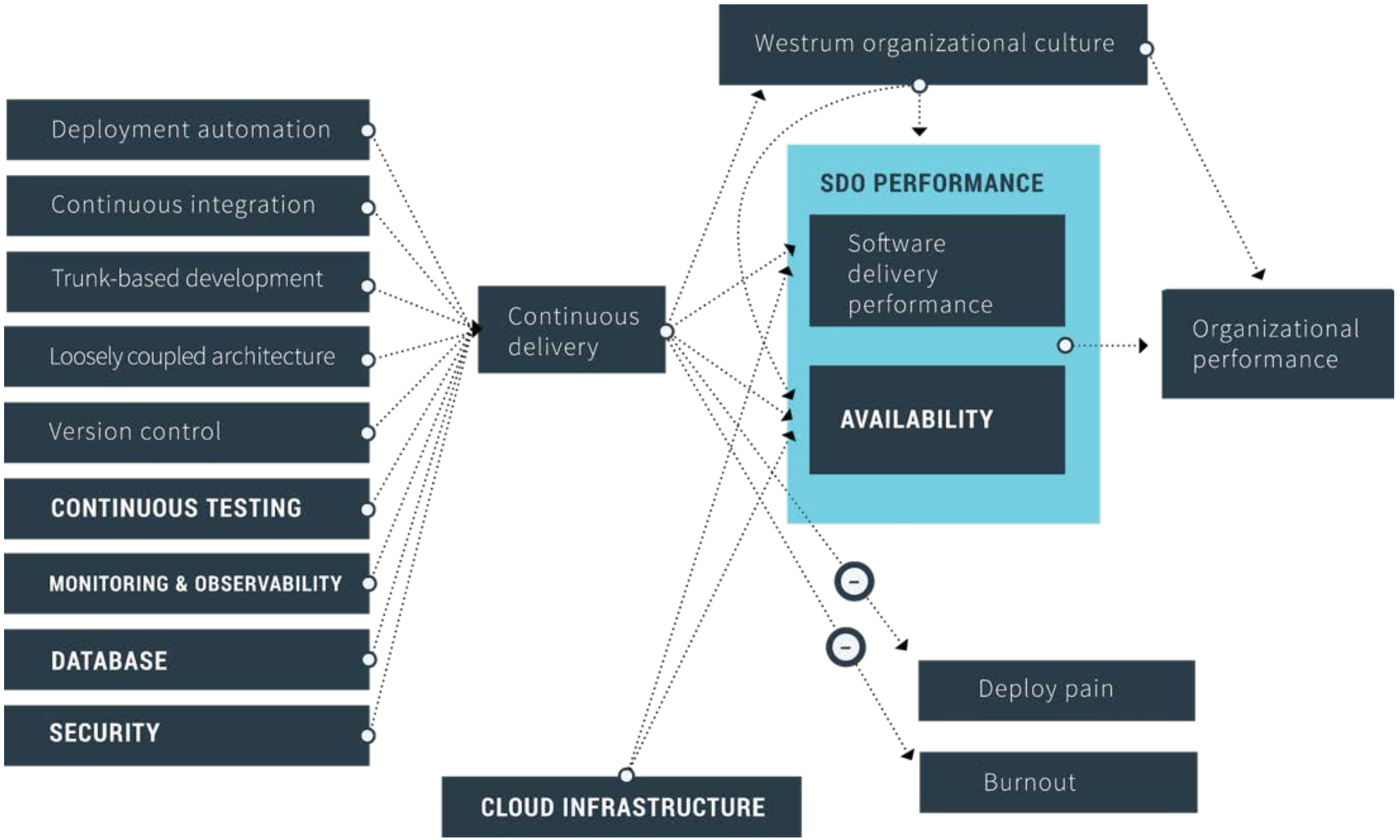
https://continuousdelivery.com/

build quality in

"Cease dependence on mass inspection to achieve quality. Improve the process and *build quality into the product* in the first place"

W. Edwards Deming

technical practices



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

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

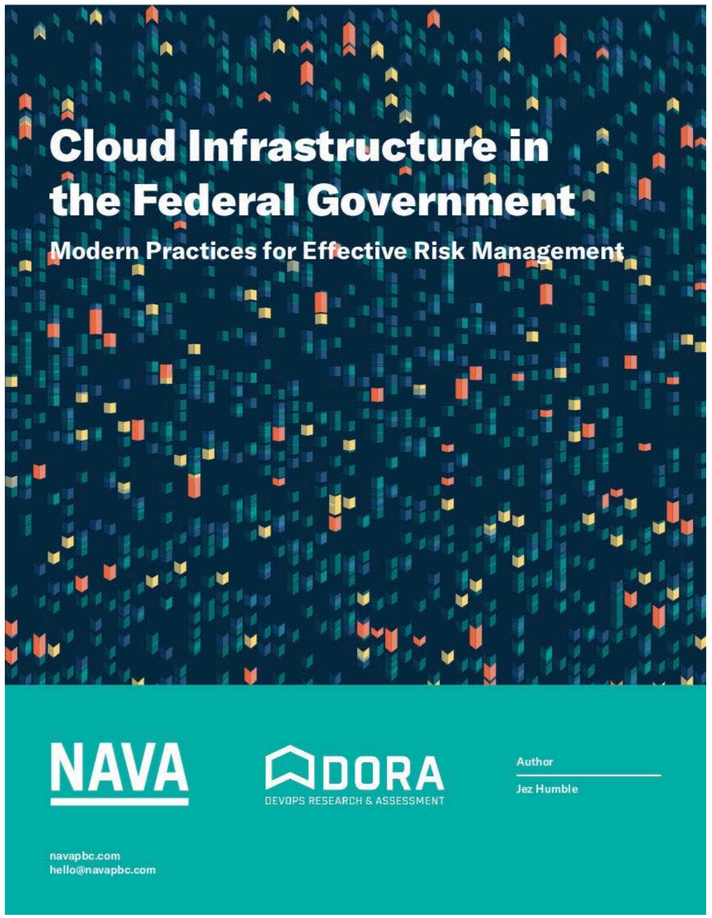
%	AGREED OR STRONGLY AGREED
46%	**On-demand self-service**
46%	**Broad network access**
43%	**Resource Pooling**
45%	**Rapid elasticity**
48%	**Measured service**

Only 22% of teams are doing cloud right!

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

http://bit.ly/2018-devops-report | NIST SP 800-145

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?

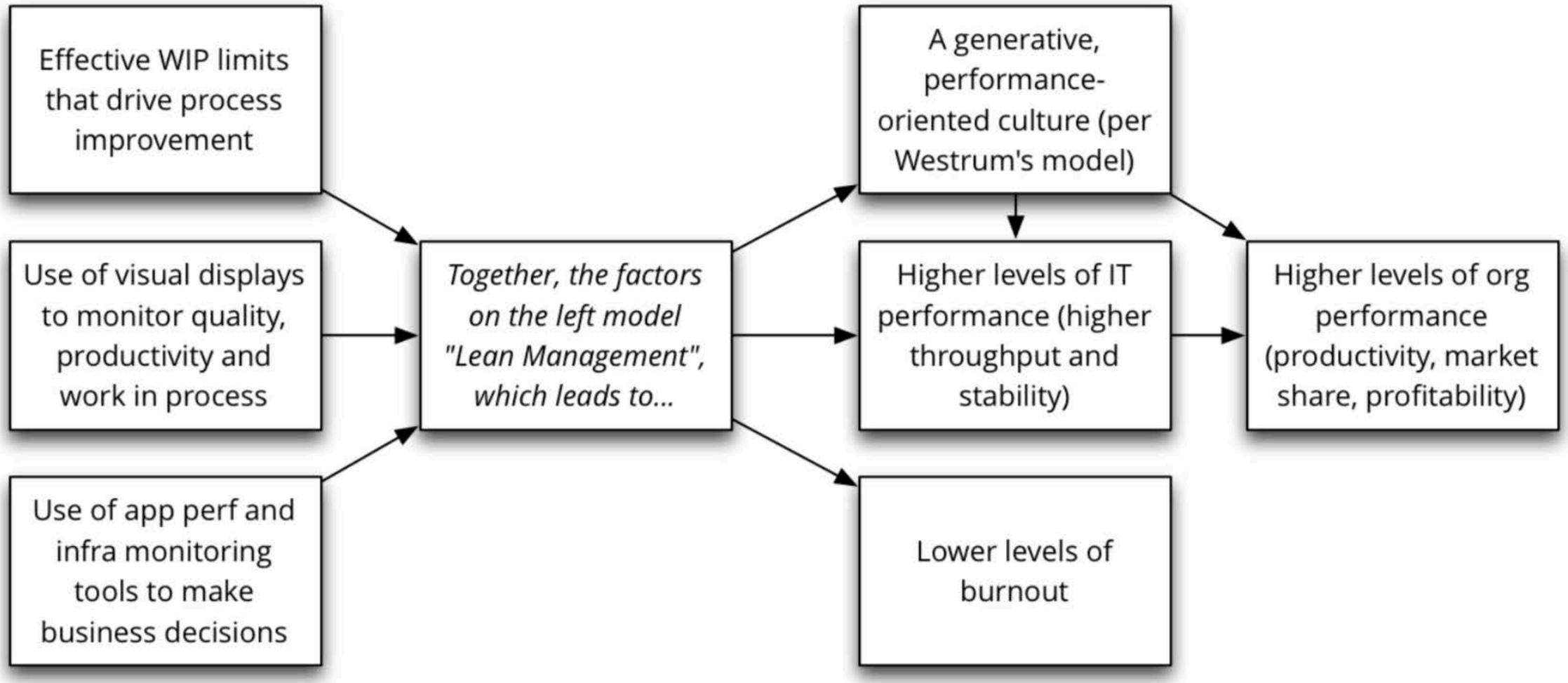
- Developers primarily create & maintain acceptance tests
- QA primarily create & maintain acceptance tests
- Primarily created & maintained by outsourced party
- When automated tests pass, I'm confident the software is releasable
- Test failures are likely to indicate a real defect
- It's easy for developers 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

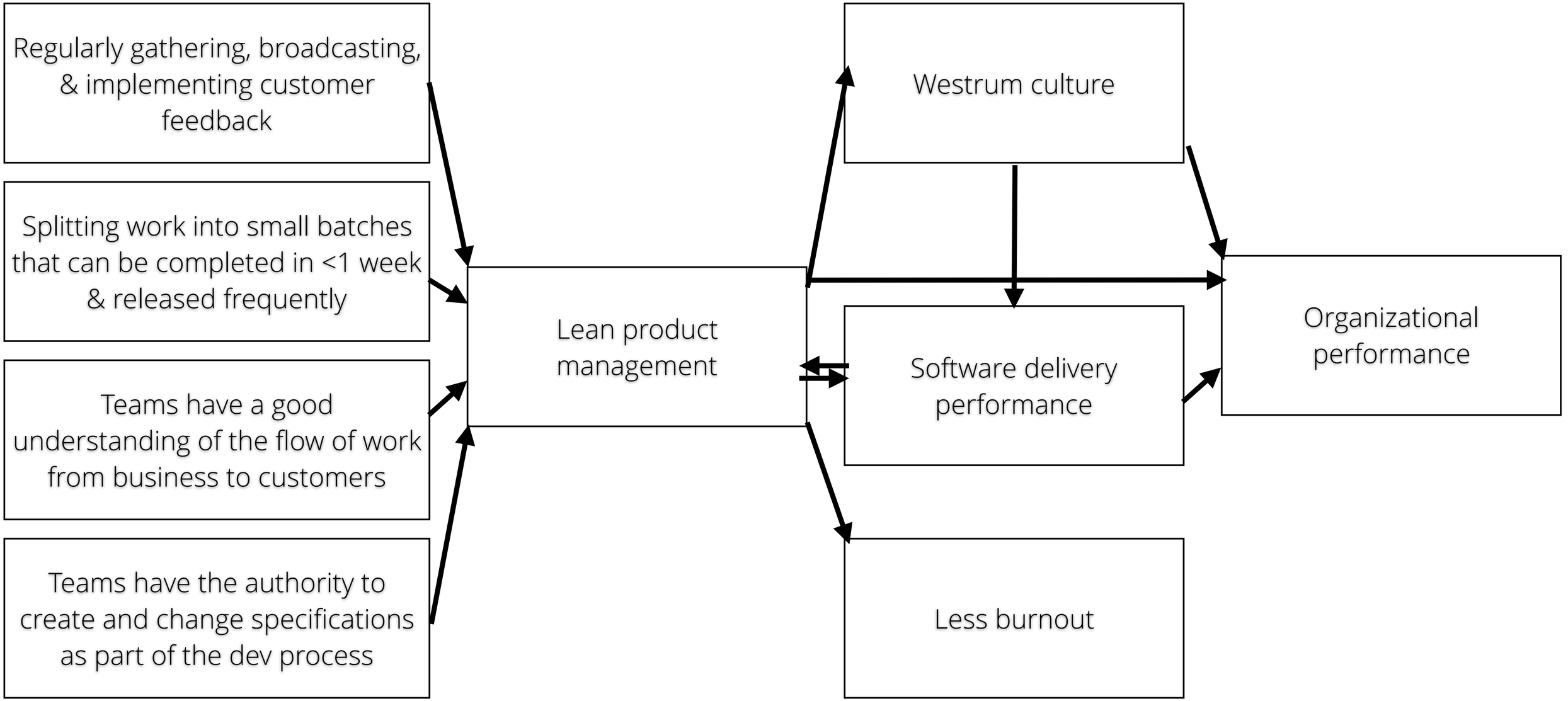
previous practices plus...

- 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

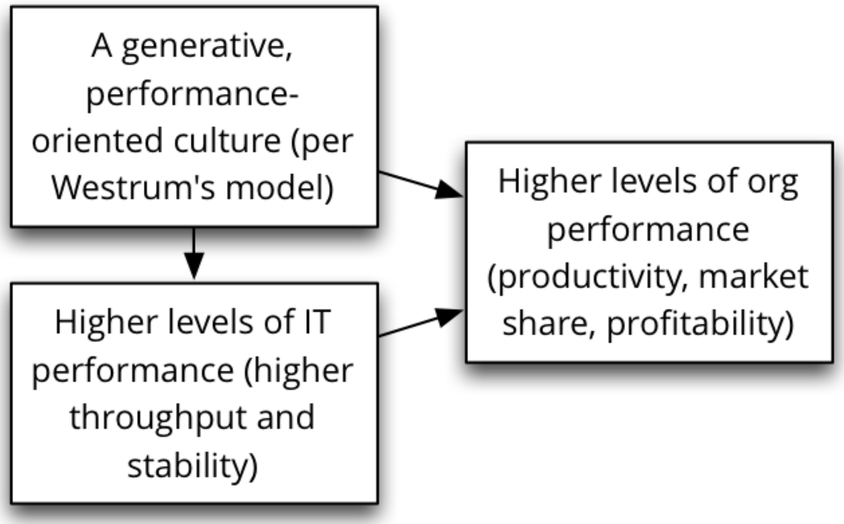
lean management



lean product management



culture impacts performance



high trust culture

how organizations process information

Pathological (*power oriented*)	Bureaucratic (*rule oriented*)	Generative (*performance oriented*)
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

Westrum, “A Typology of Organizational Cultures” | http://bmj.co/1BRGh5q

effective teams

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

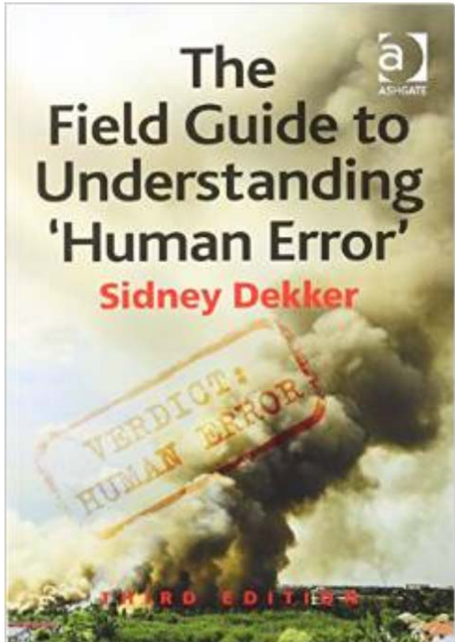
dealing with failure

in a complex, adaptive system failure is inevitable

when accidents happen, human error is the starting point of a blameless post-mortem

ask: how can we get people better information?

ask: how can we detect and limit failure modes?



The immediate response from *everyone* around was to ask, “What help do you need?”

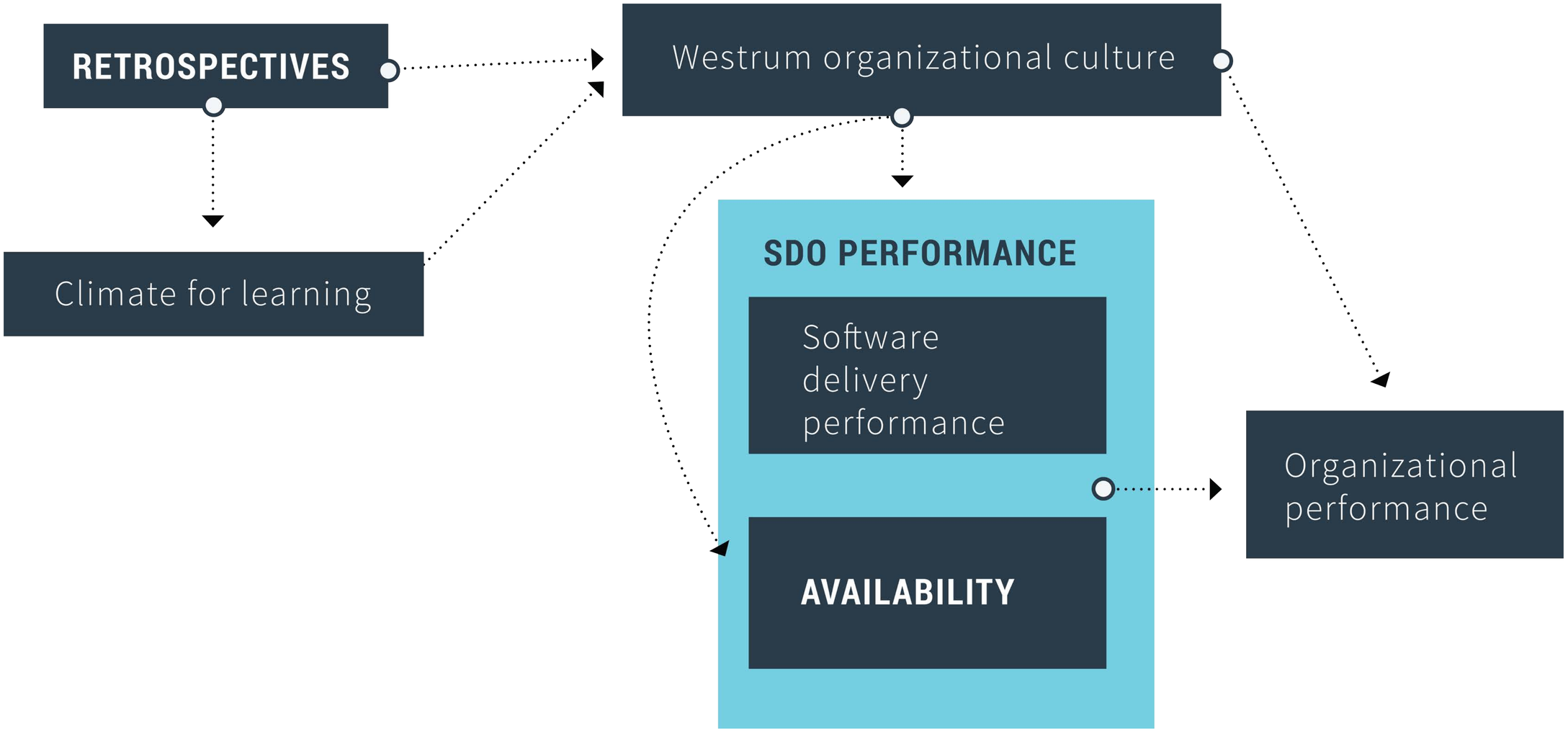
disaster recovery testing

“For DiRT-style events to be successful, an organization first needs to accept system and process failures as a means of learning... We design tests that require engineers from several groups who might not normally work together to interact with each other. That way, should a real large-scale disaster ever strike, these people will already have strong working relationships”

—Kripa Krishnan, Director, Cloud Operations, Google

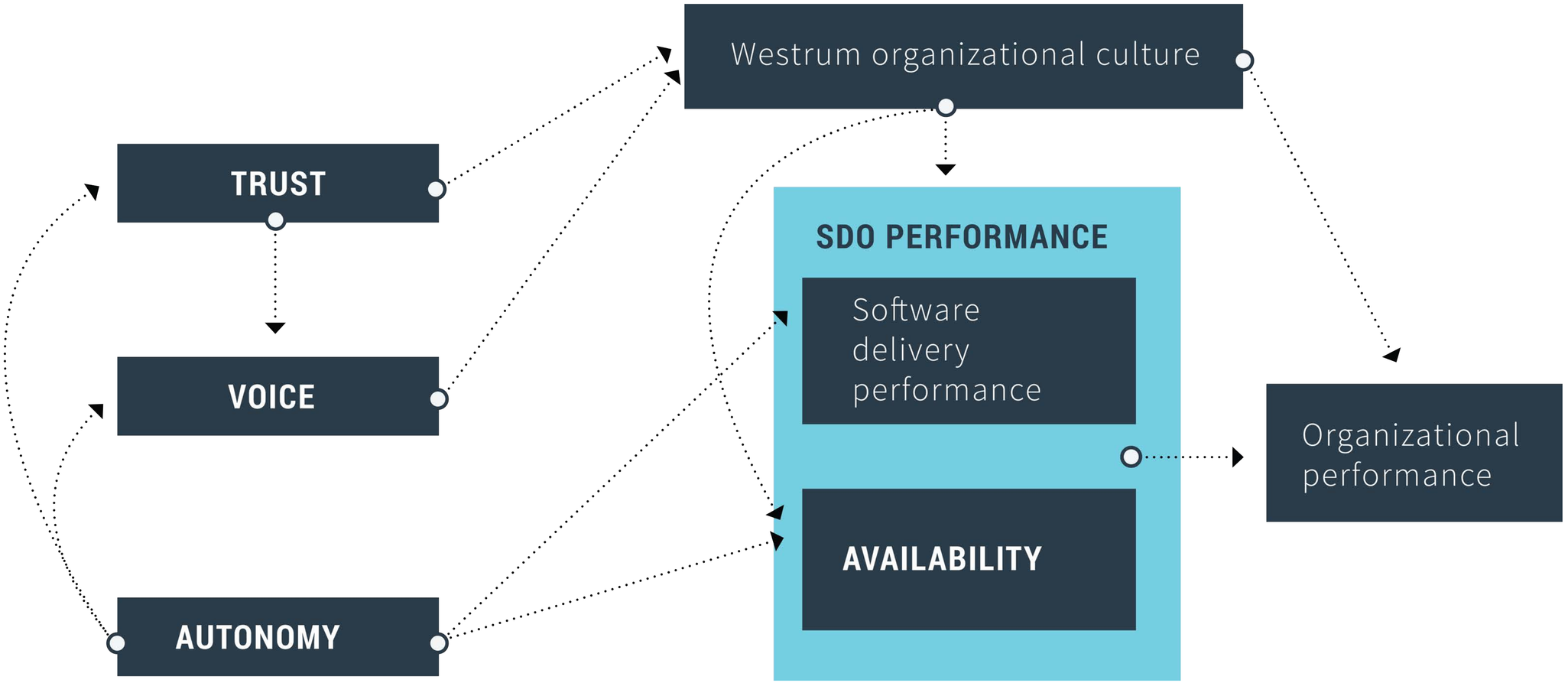
Kripa Krishnan | http://queue.acm.org/detail.cfm?id=2371297

climate for learning



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

autonomy



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

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

NETFLIX

https://www.slideshare.net/reed2001/culture-1798664/94-Highly_Aligned_Loosely_Coupled_Highly

"If it's a good idea, go ahead and do it. It is much easier to apologize than it is to get permission."

—Rear Admiral Grace Hopper, USN, 1906-1992

thank you!

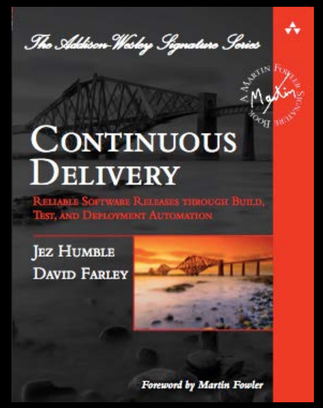
To receive the following:

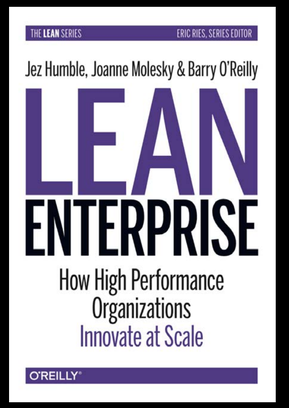
- An excerpt from *Accelerate*
- 30% off Jez's new video course: creating high performance organizations
- 50% off Jez's CD video training, interviews with Eric Ries, and more
- A copy of this presentation
- A 100 page excerpt from *Lean Enterprise*
- An excerpt from *The DevOps Handbook*
- A 20m preview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

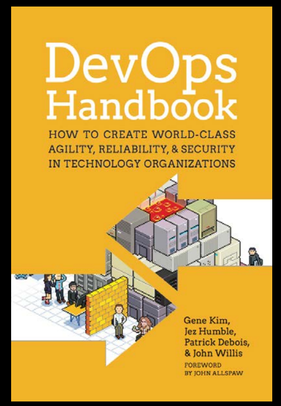
Just pick up your phone and send an email

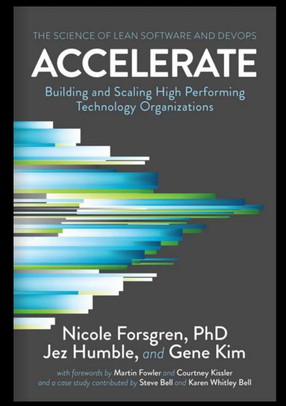
To: jezhumble@sendyourslides.com

Subject: devops









CC BY NC SA

© 2016-8 DevOps Research and Assessment LLC and Jez Humble and Associates LLC