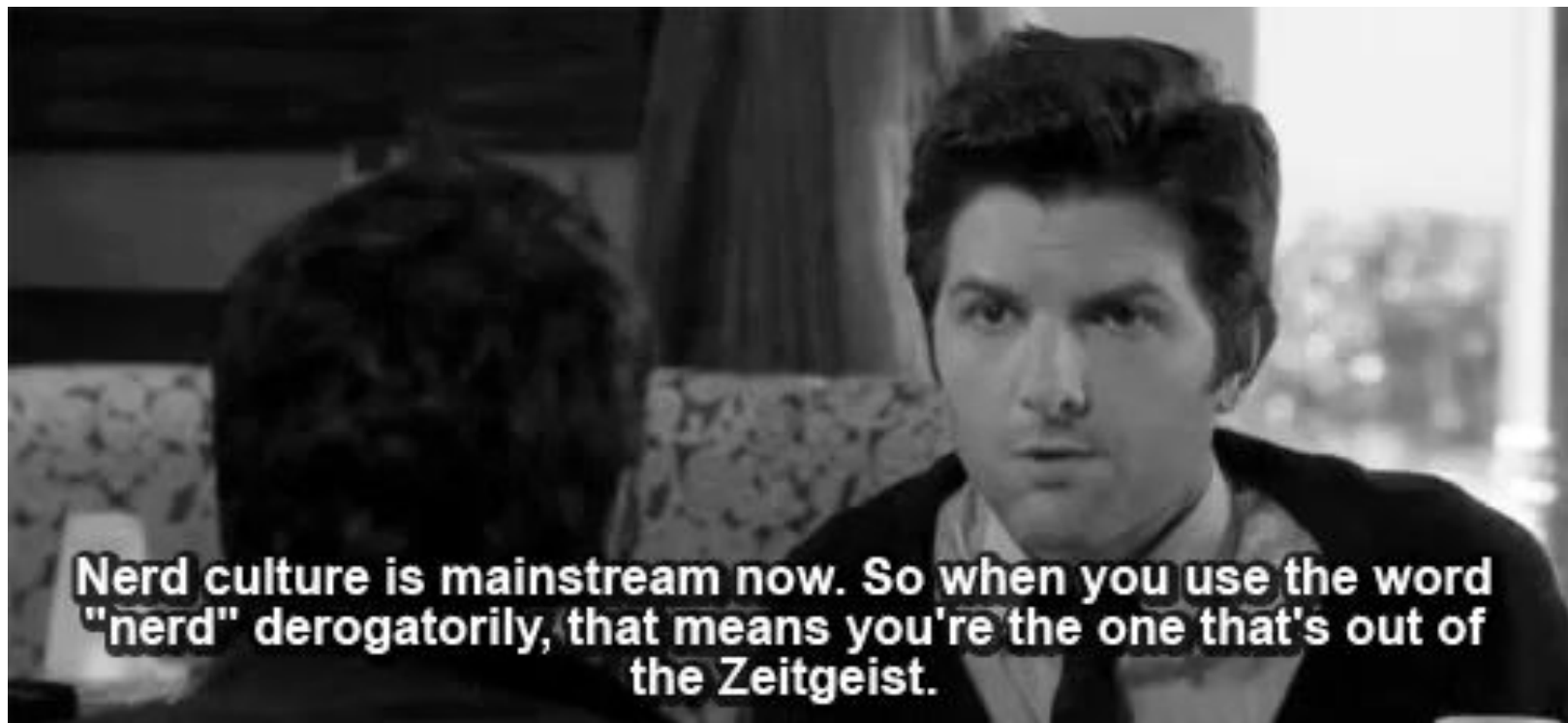


7 Things Executives Can Learn From Software Engineers

Matt Stratton

Staff Developer Advocate • Pulumi

@mattstratton



Nerd culture is mainstream now. So when you use the word "nerd" derogatorily, that means you're the one that's out of the Zeitgeist.

NS1.

The power of iteration



Iterative

1



2



3



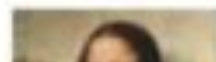
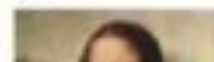
4



5



Incremental



Small teams
move fast



NS1.

Small teams move fast

- Glue work between the teams
- Black box squads (input/output)
- Conway's law
- Software contracts extend to teams (“this api does x” etc)

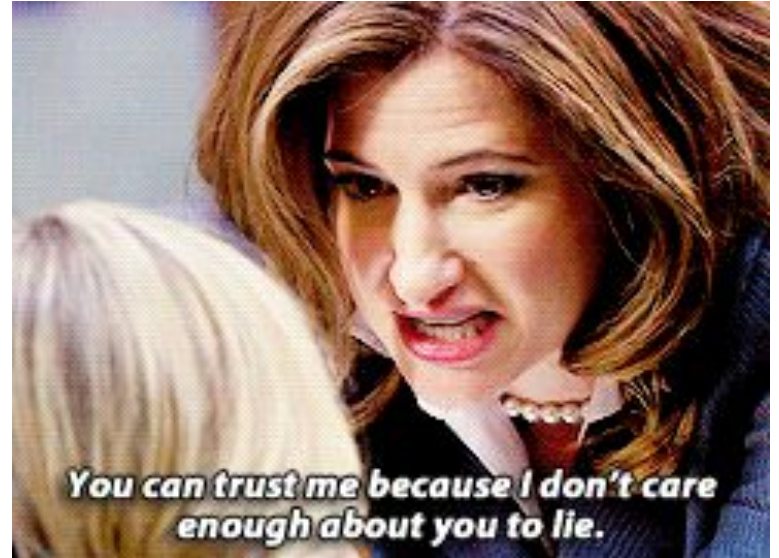
Fast feedback



Fast Feedback

- Set success criteria that is measurable
- Get changes in front of users/consumers quickly
- Create a process to share all feedback as soon as possible
- Automate feedback where possible

Trust but
verify



NS1.

Trust, But Verify

- Guardrails are good!
- Make the right way the easy way
- People want to do the right thing, but mistakes can happen

Collaboration over competition



Collaboration over Competition

- Power structures can create unintended consequences
- Ownership/fiefdoms
- “Who moved my cheese”? (or who touched it)
- Resource guarding

Westrum Model

Pathological	Bureaucratic	Generative
Power oriented	Rule oriented	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 inquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

Community spirit



No one achieves anything alone.

Community Spirit

- Developers love to work in communities
- Communities can be internal and external
- We can learn from our peers - and help them as well
- Fewer things have to be secret than we think

Learning culture



Learning Culture

- Learning from Incidents
- Shadow rotations (not just for juniors and not just for tech)
- Blamelessness



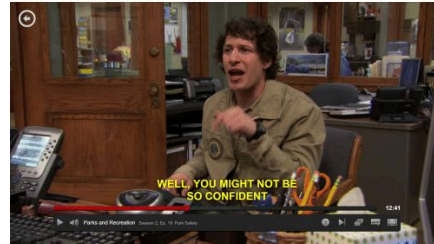
Incidents are unplanned investments; their costs have already been incurred. Your org's challenge is to get ROI on those events.

- John Allspaw, Adaptive Capacity Labs



RCA \neq learning

Shadow Rotations



The impulse to blame and punish has the unintended effect of disincentivizing the knowledge sharing required to learn from incidents



Resilience and Organizational Dynamics

NS1.

Blunt / Sharp End



Blunt End

Removed from
experience

Upstream decision
makers

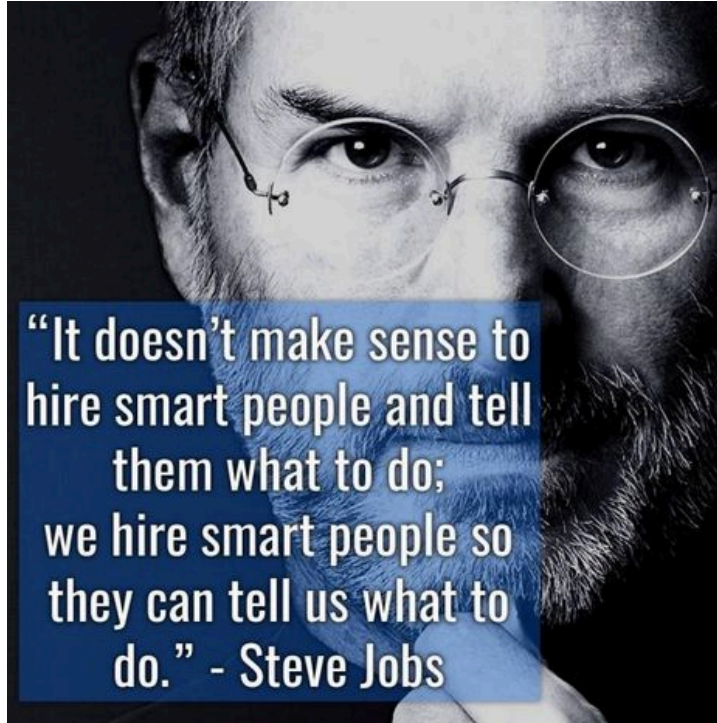


Sharp End

People directly
engaged in the
work

“Chop wood, carry
water”

Sharp End



Constantly building and destroying systems

Strong signaling

Improve systems based on strain

Will do so naturally if given ownership



[Psychological safety is] a sense of confidence that the team will not embarrass, reject, or punish someone for speaking up

Amy Edmondson, Professor, Harvard Business School



Radical candor?

If you are asking for candor/blunt feedback, what are you doing to make this safe?

Thank you!

Twitter - @mattstratton

GitHub - mattstratton

Slides - speaking.mattstratton.com

LinkedIn - linkedin.com/in/mattstratton

Podcast - ArrestedDevOps.com

DevOps Party Games - devopspartygames.com

