# Psychological Safety in Operation Teams

JOHN P. LOONEY

John Looney is a Systems Engineer at Intercom, helping to build a modern SaaS-based infrastructure platform. Before that, he was a full-stack SRE at Google, where he did everything from rack design and datacenter automation to ad-serving; he had stops at GFS, Borg, and Colossus along the way. He wrote a chapter of the SRE book on automation and is on the steering committee for USENIX SREcon. valen@tuatha.org

When I worked for Google as a Site Reliability Engineer, I was lucky enough to travel around the world with a group called "Team Development." Our mission was to design and deliver team-building courses to teams who wanted to work better together. Our work was based on research later published as Project Aristotle [1]. It found that the primary indicator of a successful team wasn't tenure, seniority, or salary levels but psychological safety.

Think of a team you work with closely. How strongly do you agree with these five statements?

1. If I take a chance and screw up, it will be held against me.
2. Our team has a strong sense of culture that can be hard for new people to join.
3. My team is slow to offer help to people who are struggling.
4. Using my unique skills and talents comes second to the objectives of the team.
5. It's uncomfortable to have open, honest conversations about our team's sensitive issues.

Teams that score high on questions like these can be deemed to be "unsafe." Unsafe to innovate, unsafe to resolve conflict, unsafe to admit they need help. Unsafe teams can deliver for short periods of time, provided they can focus on goals and ignore interpersonal problems. Eventually, unsafe teams will underperform or shatter because they resist change.

Let me highlight the impact an unsafe team can have on an individual, through the eyes of an imaginary, capable, and enthusiastic new college graduate.

This imaginary graduate, I'll call her Karen, read about a low-level locking optimization for distributed databases and realized it applied to the service her team was on-call for. Test results showed a 15% CPU saving! She excitedly rolled it out to production. Changes to the database configuration file didn't go through the usual code-review process, and, unfortunately, it caused the database to hard-lock-up. There was a brief but total Web site outage. Thankfully, her more experienced colleagues spotted the problem and rolled back the change inside of 10 minutes. Being professionals, the incident was discussed at the weekly "postmortem" meeting.

*1. "If I take a chance, and screw up, it'll be held against me."*

At the meeting, the engineering director asserted that causing downtime by chasing small optimizations was unacceptable. Karen was described as "irresponsible" in front of the team. The team suggested ways to ensure it wouldn't happen again. Unlike Karen, the director soon forgot about this interaction.

Karen would never try to innovate without explicit permission again.

*2. "Our team has a strong sense of culture, and it's hard for new people to join."*

The impact on Karen was magnified because no one stood up for her. No one pointed out the lack of code reviews on the database configuration. No one highlighted the difference between one irresponsible act and labeling someone "irresponsible." The team was proud of their system's reliability, so defending their reputation was more important than defending a new hire.

Karen learned that her team and manager didn't have her back.

*3. "My team is slow to offer help to people who are struggling."*

Karen was new to being on-call for a "production" system, so had no formal training in incident management, production hygiene, or troubleshooting distributed systems. Her team was mostly made up of people with decades of experience, who never needed training or new-hire documentation. There were no signals that it was OK for a new graduate to spend time learning these skills.

Karen was terrified of being left with the pager. She didn't understand how she passed the hiring process, and frequently wondered why she hadn't been fired yet. We call this Imposter Syndrome [2].

*4. "Using my unique skills and talents comes second to the goals of the team."*

Karen's background was in algorithms, data structures, and distributed computing. She realized the existing system had design flaws and could never handle load spikes. The team had always blamed the customers for going over their contracted rates, which is like blaming weathermen for rain during an Irish barbecue. Strong operations teams need a mix of people from different backgrounds. It's not always clear whether a problem will require understanding a database schema, Ruby debugging, C++ performance understanding, product knowledge, or people skills.

Karen proposed a new design, based on technology she'd used during her internship. Her coworkers were unfamiliar with the new technology and considered it too risky. Karen dropped her proposal without discussion. She wanted to write code and build systems, not have pointless arguments.

*5. "It's uncomfortable to have open, honest conversations about our team's sensitive issues."*

When a large customer traffic spike caused the product to be unavailable for a number of hours, the CEO demanded a meeting with the operations team. Many details were discussed, and Karen explained that the existing design meant it could never deal with such spikes and mentioned her design. Her director reminded her that her design had already been turned down at an Engineering Review and promised the CEO they could improve the existing design.

Karen discussed the meeting with one of her teammates afterwards. She expressed dismay that the director couldn't see that his design was the root-cause of their problems. The teammate shrugged and pointed out that the team had delivered a really good service for the last five years and had no interest in arguing about alternate designs with the director.

Karen left work early to look for a new job. The company didn't miss her when she left. After all, she was "reckless, whiny and had a problem with authority." They didn't reflect on the design that would have saved the company from repeated outages that caused a customer exodus.

## How to Build Psychological Safety into Your Own Team

What is special about Operations that drives away so many promising engineers and suffers others to achieve less than their potential?

We know that success requires a strong sense of culture, shared understandings and common values. We have to balance that respect for our culture with an openness to change it as needed. A team—initially happy to work from home—needs to co-locate if they take on interns. Teams—proud that every engineer is on-call for their service—may need to professionalize around a smaller team of operations-focused engineers as the potential production impact of an outage grows.

We need to be thoughtful about how we balance work people love with work the company needs to get done. Good managers are proactive about transferring out an engineer who is a poor fit for their team's workload. Great managers expand their team's remit to make better use of the engineers they have, so they feel their skills and talents are valued. Engineers whose skills go unused grow frustrated. Engineers ill-equipped to succeed at assigned work will feel set up to fail.

### *Make Respect Part of Your Team's Culture*

It's hard to give 100% if you spend mental energy pretending to be someone else. We need to make sure people can be themselves by ensuring we say something when we witness disrespect. David Morrison (Australia's Chief of the Army) captured this sentiment perfectly in his "the standard you walk past is the standard you accept" [3] speech. Being thoughtless about people's feelings and experiences can shut them down. Some examples where I've personally intervened:

- Someone welcomes a new female project manager to the team, assumes they aren't technical, and uses baby words to explain a service. I highlight the new PM has a PhD in CS. No harm was intended, and the speaker was mortified that their good-humored introduction was inappropriate.
- In a conversation about people's previous positions, someone mentioned they worked for a no-longer-successful company, and a teammate mocked them for being "brave enough" to admit it. I pointed out that mocking people is unprofessional and unwelcome, and everyone present understood a "line" that hadn't been visible previously.
- A quiet, bright engineer consistently gets talked over by extroverts in meetings. I point out to the "loud" people that we were missing an important viewpoint by not ensuring everyone speaks up. Everyone becomes more self-aware.

It's essential to challenge lack of respect immediately, politely, and in front of everyone who heard the disrespect. It would have been wonderful had someone reminded Karen's director, in front of the group, that Karen wasn't irresponsible, the outage wasn't a big deal, and the team should improve their test coverage.

### *Make Space for People to Take Chances*

Some companies talk of 20% time. Intercom, where I work, has "buffer" weeks, in between some of our six-week sprints [4]. People often take that chance to scratch an itch that was bothering them, without impacting the external commitments the team has made. Creating an expectation that everyone on the team has permission to innovate, and encouraging the whole team to go off-piste at the same time, sends a powerful message.

Be careful that "innovation time" isn't the only time people should take chances. I've worked with one company in the car industry that considers "innovation time" to be 2:30 p.m. on Tuesdays!

Imagine how grateful Karen would have been had a senior engineer at the Engineering Review offered to work on her design with her so that it was more acceptable to the team. Improve people's ideas rather than discounting them.

### *Make It Obvious When Your Team Is Doing Well*

One engineer describes his experience of on-call as "being like the maintenance crew at the fairground. No one notices our work, until there is a horrible accident." Make sure people notice when your team is succeeding.

I love how my team writes goals on Post-It notes at our daily standups and weekly goal meetings. These visible marks of success can be cheered as they are moved to the "done" pile. But we can also celebrate glorious failure.

Many years ago, when I was running one of Google's storage SRE teams, we were halfway through a three-year project to replace the old Google File System. Through a confluence of bad batteries, firmware bugs, poor tooling, untested software, an aggressive rollout schedule, and two power cuts, we lost a whole storage cell for a number of hours. Though all services would have had storage in other availability zones, the team spent three long days and three long nights rebuilding the cluster. Once it was done, they—and I—were dejected. Demoralized. Defeated. An amazing manager (who happened to be visiting our office) realized I was down, and pointed out that we'd just learned more about our new storage stack in those three days than we had in the previous three months. He reckoned a celebration was in order.

I bought some cheap sparkling wine from the local supermarket and, with another manager, took over a big conference room for a few hours. Each time someone wrote something they learned on the whiteboard, we toasted them. The team that left that room was utterly different from the one that entered it.

I'm sure Karen would have loved appreciation for her uncovering the team's weak non-code test coverage and their undocumented love of uptime-above-all-else.

### *Make Your Communication Clear and Your Expectations Explicit*

Rather than yelling at an engineering team each time they have an outage, help them build tools to measure what an outage is, a Service Level Objective that shows how they are doing, and a culture that means they use the space between their objective and reality to choose to do the most impactful work.

When discussing failures, people need to feel safe to share all relevant information, with the understanding that they will be judged not on how they fail, but how their handling of failures improved the team, their product, and the organization as a whole. Teams with operational responsibilities need to come together and discuss outages and process failures. It's essential to approach these as fun learning opportunities, not root-cause-obsessed witch-hunts.

I've seen a team paralyzed, trying to decide whether to ship an efficiency win that would increase end-user latency by 20%. A short conversation with the product team resulted in updates to the SLO, detailing "estimated customer attrition due to different latency levels," and the impact that would have on the company's bottom line. Anyone on the team could see in seconds that low-latency was far more important than hardware costs and instead drastically over-provisioned.

If you expect someone to do something for you, ask for a specific commitment—"When might this be done?"—rather than assuming everyone agrees on its urgency. Trust can be destroyed by missed commitments.

Karen would have enjoyed a manager who told her in advance that the team considered reliability sacred and asked her to work on reliability improvements rather than optimizations.

### *Make Your Team Feel Safe*

If you are inspired to make your team feel more psychologically safe, there are a few things you can do today:

1. Give your team a short survey (like the questions listed above), and share the results with your team.
2. Discuss what "safety" means to your team; see if they'll share when they felt "unsafe."
3. Build a culture of respect and clear communication, starting with your actions.

Treat psychological safety as a key business metric, as important as revenue, cost of sales, or uptime. This will feed into your team's effectiveness, productivity, staff retention, and any other business metric you value.

## Why Are Operations Teams More Likely to Feel Unsafe than Other Engineering Teams?

### *We Love Interrupts and Information*

Humans suck at multitasking. Trying to do multiple things at once either doubles the time the task takes or doubles the mistakes [5]. A team that's expected to make progress with project work while being expected to be available for interrupt work (tickets, on-call, walkups) is destined to fail. And yet, operations attracts people who like being distracted by novel events. Do one thing at a time. Timebox inbound communications as well as interrupt time.

Operations teams are expected to manage risk and uncertainty for their organization. We build philosophies for reasoning about risk and strategies for coping with bad outcomes, defense in depth, playbooks, incident management, escalation policies, etc. When humans are exposed to uncertainty, the resultant "information gap" results in a hunger for information, often exaggerated past the point of utility [6]. This can lead to information overload in the shape of ludicrously ornate and hard to understand dashboards, torrents of email, alerts, and automatically filed bugs. We all know engineers who have hundreds of bugs assigned to them, which they cannot possibly ever fix, but refuse to mark them "Won't Fix." Another pathology is subscribing to developer mailing lists to be aware of every change being made to the system. Our love of novelty blinds us to the lack of value in information we cannot act on.

Admit that most information is not actionable, and be brutal with your bugs, your mail filters, and your open chat apps.

### *On-Call and Operations*

The stress of on-call is what drives people away from operations roles. Curiously, 24/7 shifts are not the problem. The real problem is small on-call rotations that result in long, frequent shifts. The more time people spend on-call, the more likely they are to suffer from depression and anxiety [7]. The expectation of having to act is more stressful than acting itself [8]. It's one thing to accept that on-call is part of a job. It's another to tell your five-year-old daughter you can't bring her to the playground.

We can mitigate this stress by ensuring on-call rotations of no fewer than six people, with time-in-lieu for those with significant expectations around response times, or personal life curtailment. Compensate based on time expecting work, not time doing work. Incident management training or frequent "Wheel of Misfortune" drills can also reduce stress, by increasing people's confidence. Ensure on-call engineers prioritize finding someone to fix a problem when multiple incidents happen concurrently [9].

### *Cognitive Overload*

Operations teams support software written by much larger teams. I know a team of 65 SREs that supports software written by 3,500 software engineers. Teams faced with supporting software written in multiple languages, with different underlying technologies and frameworks spend a huge amount of time trying to understand the system and so have less time to improve it.

To reduce complexity, software engineers deploy more and more abstractions. Abstractions can be like quicksand. ORM (object-relational mapping) [10] is a wonderful example of a tool that can make a developer's life easy by reducing the amount of time thinking about database schemas. By obviating the need for developers to understand the underlying schema, developers no longer consider how ORM changes impact production performance. Operations now need to understand the ORM layer *and* why it impacts the database.

Monolithic designs are often easier to develop and extend than microservices. There can be valid business reasons to avoid duplication of sensitive or complex code. However, because they attract heterogeneous traffic classes and costs, they are a nightmare for operations teams to troubleshoot or capacity plan.

Everyone understands that onboarding of new, evolving software strains an operations team. We ignore the burden of mature "stable" services. There is rarely any glamorous work to be done on such services, but the team still needs to understand it. Mature services can silently swamp an operations team.

Ensure teams document the impact of cognitive load on development velocity. It has a direct and serious impact on the reliability of the software, the morale and well-being of the operations team, and the long-term success of the organization.

### *Imaginary Expectations*

Good operations teams take pride in their work. When there is ambiguity around expectations of a service, we will err on the side of caution and do more work than needed. Do we consider all of our services to be equally important? Are there some we can drop to "best effort"? Do we really have to fix all bugs logged against our team, or can we say, "Sorry, that's not our team's focus"? Are our SLAs worded well enough that the entire team knows where their effort is best directed on any given day? Do we start our team meeting with the team's most important topics, or do we blindly follow process?

Ensure there are no magic numbers in your alerts and SLAs; if your team is being held to account for something, ensure there is a good reason that everyone understands.

### *Operations Teams Are Bad at Estimating Their Level of Psychological Safety*

Lastly, I'll leave you with a thought: *people who are good at operations are bad at recognizing psychologically unsafe situations*. We consider occasionally stressful on-call "normal" and don't feel it getting worse until we burn out. The curiosity that allows us to be creative drives us to information overload. Despite being realistic about how terrible everything is, we stay strongly optimistic that the systems, software, and people we work with will get better.

I've given surveys to deeply troubled teams where every response seemed to indicate everything was wonderful. I'd love to hear from people who have experience uncovering such cognitive dissonance in engineers.

### *References*

[1] J. Rozovsky, "Five Keys to a Successful Google Team": http://bit.ly/1X0Uygj.

[2] Wikipedia, "Imposter Syndrome," last edited 9/21/17: https://en.wikipedia.org/wiki/Impostor_syndrome.

[3] D. Morrison speech transcript: http://bit.ly/2fkDqnu.

[4] Intercom blog, "6 Weeks: Why It's the Goldilocks of Product Timeframes": https://blog.intercom.com/6-week-cycle-for-product-teams/.

[5] P. Atchley, "You Can't Multitask, So Stop Trying," *Harvard Business Review*, Dec 21, 2010: https://hbr.org/2010/12/you-cant-multi-task-so-stop-tr.

[6] G. Loewenstein, "The Psychology of Curiosity," *Psychological Bulletin*, vol. 116, no. 1, 1994: http://bit.ly/2xmqpOE.

[7] A.-M. Nicol and J. S. Botterill, "On-Call Work and Health: A Review," *Environ Health*, vol. 3, 2004: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC539298/.

[8] J. Dettmers, T. Vahle-Hinz, E. Bamberg, N. Friedrich, M. Keller, "Extended Work Availability and Its Relation with Start-of-Day Mood and Cortisol," *Journal of Occupational Health Psychology*, vol. 21, no. 1, Jan. 2016: https://www.ncbi.nlm.nih.gov/pubmed/26236956.

[9] D. O'Connor, "Bad Machinery: Managing Interrupts under Load," SREcon15 Europe, USENIX: http://bit.ly/2xWjbnZ.

[10] Wikipedia, "Object-Relational Mapping," last edited 7/7/17: https://en.wikipedia.org/wiki/Object-relational_mapping.