The Five Stages of SRE

BEN PURGASON

Ben Purgason is a Director of Site Reliability Engineering at LinkedIn, responsible for the operational integrity of LinkedIn's internal software development, trust, and security infrastructure. In his six years at LinkedIn, he's developed seven successful SRE teams that partner with more than 21 distinct teams to build reliability into their entire software life cycle. bpurgason@linkedin.com

I am a rebel in a world that made development vs. operations the status quo. I am an SRE leader with LinkedIn, and I'd like to share with you my observations and experiences building and making Site Reliability Engineering (SRE) teams successful there. In particular, I want to share a few key observations on the evolutionary path of SRE, how that path is influenced by the relationship with the software engineer (SWE), and how the allocation of roles and responsibilities changes with time.

Founding Principles

Back in 2010, SRE was founded at LinkedIn as a scrappy band of firefighters. Their first job was to put out the blazing operational fires that threatened LinkedIn's stability on a daily basis. I joined two years later in 2012, and though progress had been made, the environment was still chaotic. Putting things in perspective, when I joined the company, all on-calls were expected to be on-site by 6 a.m. each day. At the time, most of LinkedIn's traffic came from the Americas, which resulted in a drastic traffic increase as the continents woke up, usually knocking over the site. In spite of the chaos, or perhaps because of it, three fundamental principles were created to guide the development of the SRE organization.

The first is **Site Up** [1]. This is our highest technical priority—always. Site Up is ensuring that the site, service, experience, app, etc. you offer to customers work correctly and quickly enough to be relevant. Without Site Up you cannot uphold the commitments you made to your customers. If you cannot uphold your commitments, your users will eventually lose their trust in your company and will take their business elsewhere. Let this go on long enough and eventually everyone at your company (including you) will be out on the street looking for a new job.

The second is **Empower Developer Ownership**. As Bruno Connelly, head of SRE at LinkedIn, likes to say: "It takes a village to run a website." In other words, it is crucial that developers own the Operations problem as much as everyone else. Operations is a hard problem, and we're going to do a better job solving it if we bring 100 percent of our engineering talent to bear rather than just the 10 percent that happens to have "site reliability" in their title.

The third is **Operations Is an Engineering Problem**. We think Operations should be solved like any other software problem: by engineering permanent solutions to problems or, better, preventing them from happening in the first place. We do not want to solve our problems through heroes and raw effort.

As previously mentioned, Site Up is our chief technical operating priority, and nothing will ever beat it. This introduces a challenge, for Site Up is a far-reaching concept that can lead us down a short-sighted and dangerous path if we aren't careful. The other two principles, "Empower Developer Ownership" and "Operations Is an Engineering Problem," are both constraints. They ensure that our efforts to uphold Site Up are sustainable today and become increasingly effective with time.

Generation 1: The Firefighter

Let's face it, few companies invest in creating an SRE organization until either they have a raging operational fire on their hands or the minimum change velocity—that's the minimum amount of change that must occur each day for the company to still remain competitive—exceeds what traditional operations processes can handle.

At this early stage, SRE is essentially fighting fires whenever the need arises while simultaneously trying to automate the process of fire suppression. With each piece of reliable automation written, the time saved on fire suppression is freed up for use on permanently fixing problems or for forward-looking priorities such as monitoring and alerting.

Though the focus is on Site Up (incident management), we can also see the earliest influences of Operations Is an Engineering Problem through the automation of manual operational work.

Tools SRE, the Firefighters

Tools SRE, the team I founded at LinkedIn, is responsible for all SRE-related tasks around our internal Tooling (development, build, and deploy pipelines), along with those around Trust and Security.

When Tools SRE was founded, the mean time to resolve an incident (MTTR) was over 1500 minutes (yes, that is more than a full 24-hour day). So frequent and long were the outages that if we lined them all up end-to-end we would have had some level of outage every second of every day for the entire calendar year. Worse, for just under half the year, two would have been active at the same time.

So what did we do? We got after it and began our pursuit of Site Up. After every outage we held a blameless postmortem with our partner SWEs. We learned from our collective mistakes. We did our best and we didn't give up.

We instrumented our products as well as we could, and when we couldn't, we developed external observers to generate the missing metrics. We wrote alerts, we got woken up in the middle of the night, and we kept solving problems.

Be Relentless and Measure Success

Winston Churchill has been quoted, or misquoted, over the years as saying: "If you're going through hell, keep going." In a nutshell, that is the theme behind moving past this dysfunctional stage. Digging out of this particular hole takes time, patience, and an unyielding determination to succeed, but it isn't rocket science.

First, understand that **every day is Monday in Operations** [2]. There will never be an end to the problems we need to solve. In spite of that, we must continuously identify the biggest, most impactful problems and solve them. Eventually, we end up with a collection of problems that are roughly the same in quantity but greatly reduced in level of impact. Gone will be the days when developers go home early after being unable to get a build for the better part of six hours. Now you'll have users complaining about a six-minute delay in getting their build. The problems don't go away, but they do shrink in size.

Second, **what gets measured gets fixed** [3]. "What gets measured gets fixed" is a famous adage taken from a company that knew a thing or two about measurements: Hewlett Packard. Long before printers, computers, and the Internet, HP built test and measurement devices that would be more at home in a scientific laboratory than in a tech company (think oscilloscopes). They knew what they were doing—if you can measure something, you can reason about it, understand it, discuss it, and act upon it with confidence. If you cannot measure something, not only are you unable to fix it, you're unable to really understand if it's even broken or not.

If you can do just these two things, you will eventually achieve improved site stability, have more time available to invest in the future, and have a better understanding of where the next set of problems are hiding.

Generation 2: The Gatekeeper

A quick disclaimer: the gatekeeper is an evolutionary dead end that can be (almost) entirely bypassed. I include it here not to show it as the next logical step in SRE's evolution but, instead, to help any SRE team that finds itself already in it to grow past it.

As we grow past the first generation of firefighters, we achieve a basic level of operational integrity. The active operational fires are put out, and we have an increasing amount of time available for forward-looking tasks. So where do we go from here?

The natural instinct is a protective one. We just spent years digging ourselves out of a major operational hole and the last thing we want to do is immediately fall back into it. Usually this translates into a set of behaviors that amount to building a wall around "our" production with a few locked doors and keeping the keys with SRE. That way, "those" pesky developers will have to go through "us" before "their" change can impact "our" prod.

The thought is straightforward, and not entirely without merit if I'm being honest. It's motivated by fear, the fear of being in pain again.

Avoid Creating an "Us" vs. "Them" Culture

The problem with this mindset is that it quickly cements itself as a culture of "us vs. them." This is incredibly dangerous to the development of any SRE team because it will essentially limit your ability to contribute to the company and that of your SWE partners as well.

During this stage it is common for SREs to leverage their role power to claim ownership of production deployments or more generally change control. In doing so we add a new job responsibility to our SWE counterparts: get past SRE gatekeeping in the most efficient way possible.

Tools SRE, Never the Gatekeepers

The Tools SRE team I lead managed to skip this stage almost entirely. It wasn't because we were more creative or had a unique vision—it's because we got started late. When Tools SRE was founded we were outnumbered 41:1 (SWE:SRE). As a result, we knew immediately we couldn't succeed alone. Worse, any attempts at human gatekeeping would likely be ineffective—we'd just get overrun by the SWEs even if we tried to use our role power.

However, we did get a few requests for features (we were Tools SRE, after all) that directly supported human gatekeeping from other teams. Most notably was one called "service guard," which let SREs create a whitelist of individuals who could run deployments for a particular product. Essentially, this was a feature designed to force untrusted individuals to route through trusted individuals in order to do their jobs. This feature accomplished its tactical purpose in stopping unapproved deployments but also greatly increased the friction between teams, reducing their ability to collaborate.

Tribalism Has No Place in an Effective SRE Team

A man far wiser than I, Fred Kofman, has said many times that "There is no such thing as 'the hole is in your side of the boat.'" The moment you accept that your success is unachievable without your SWE partners and vice versa, you have started to grow past this generation and have begun to realize your potential. To accomplish this, you only have to do two things.

First, you must **attack the problem, not the person** [4]. Remember, your problem is neither the developers nor their changes. Your problem is Site Up. Do not attack people, they're just trying to do their jobs as best they can. Help them do their jobs better *while* supporting Site Up.

Second, we have to remember that Operations Is an Engineering Problem. If we accept that as an axiom, then we must reject the notion of human gatekeepers as the norm. Remember, we don't want to solve the operational problem using manual effort. We want to solve it as any other software problem, that is, by improving the software.

This isn't to say we don't need gatekeeping, we just don't want human gatekeepers. The second principle introduced earlier in this paper, Empower Developer Ownership, now begins to markedly influence our work.

To get rid of human gatekeeping, we need to mutually agree on what the acceptable standards are for change. Once we have an agreement, we can build automation that enforces the standards we agreed to with our SWEs. This empowers SWEs to do their job without interference, assuming the standards are met.

Let's look at how this might be applied to deployments. If an engineer is told they can't deploy their build, they're going to be unhappy. If a human tells them their build is too slow, they resent the (human) SRE, saying "They won't let me deploy." If, instead, the deployment system tells an engineer they can't deploy because their build violates the agreed upon standards, the engineer will say, "I need to make my build faster so that I can deploy." You've effectively turned a very human problem, "They won't let me deploy," into a simple engineering conversation: "I need to improve the performance of this build so that I can deploy." Well done.

As you near the exit from gatekeeper, all three core principles are now clearly on display: Site Up, Empower Developer Ownership, and Operations Is an Engineering Problem.

Generation 3: The Advocate

The advocate SREs are the ones who lay the foundational relationships required to collaborate well with our partner teams at scale. Their biggest value add is the repair and rebuilding of trusted relationships damaged during the firefighter and gatekeeper generations. How do they accomplish such a feat? They uphold the original three founding principles through their engineering solutions and their interactions with others.

Finally, we can see the list of roles and responsibilities beginning to converge with "monitoring and alerting" now appearing on both lists. By using mutually agreed upon data as a gatekeeper, both SWE and SRE end up losing something significant if the signal quality provided by the gatekeeping data degraded.

Tools SRE, the Advocates

Tools SRE was founded years after our SWE counterparts. That meant we had years of code previously written in order to keep up with the needs of a company that was rapidly growing its business and its engineering body. Reliability was difficult to achieve. Remember, we were also outnumbered 41:1. Even if we tried to use human gatekeeping, it would have failed—what good is building a gate when you don't have the resources to build a fence?

Instead, we tried a different approach. We explained that we didn't want to hold anyone back. To the contrary, we wanted to empower every engineer to do more and spend less of their time fighting fires. Over time, we refined the pitch until it came down to just a few sentences: "Look, we need your help to ensure the products built are reliable and scalable. You can either spend your time helping us fix the outages as they happen or you can create new features. Which do you want it to be?"

Trust Is Everything

As Jeff Weiner, CEO of LinkedIn, has said, "Consistency over time equals trust." This generation of SRE is all about rebuilding trusted relationships by consistently propagating Site Up culture and through building trusted relationships.

First: **be an advocate, make an advocate**. In every conversation or interaction, make sure everyone understands why Site Up matters to them. Be relentless in making this point. At a minimum, they'll eventually agree to help you because of the benefits you've attributed to Site Up and, possibly, even because they believe in the concept. Either way, once the benefits begin to appear, they'll become advocates themselves. Your job will get a bit easier as you end up with an increasingly large number of advocates.

Second: **do not insulate, share pain**. Both the firefighter and the gatekeeper tend to insulate their partner teams from pain indirectly. When the firefighters enter the scene, they help shoulder the burden of incident management. When gatekeepers build their wall, the only pain felt is that of the SREs' change-management process. If both groups are in pain, you can expect easy commitments to end the suffering of both. If only one group is in severe pain and the other feels none, you can expect only disagreements.

Generation 4: The Partner

From this point forward, SRE and SWE need to increasingly function as a single logical unit to do the most possible good. The first step in this alignment is to ensure both SWE and SRE have an equal level of dissatisfaction with the current state of reliability. Another good starting point is to begin joint SRE-SWE planning, if you haven't already. This provides a chance for mutual understanding and will serve to prevent the bulk of mid-quarter priority misalignments.

At first, your team won't be involved in every project; not every SWE team will want to play ball, and that is entirely OK. As the planning cycles go by, it'll become obvious that projects that had both SWE and SRE funding were more reliable, easier to maintain, consumed less time due to scaling problems, and were generally more successful. No one likes missing out on a competitive advantage, and any holdouts will be banging down your door demanding SRE engagement. Once this happens, your planning process gets much easier. You don't need to spend as much time trying to get involved with projects, you just have to agree to work on projects that are going to matter most to your company. Even better, when there are too many "important projects that need SRE partnership," you can go together with your SWE team to justify increased head count, since the value add is apparent to everyone in the conversation.

The roles and responsibilities of SWE-SRE are now beginning to converge rapidly, with many responsibilities being the same. One notable call out: SWE is no longer "an escalation point for SRE." Instead, both groups command a strong understanding of the code base, enough so that you may have a single hybrid on-call rotation comprising both SWE and SRE. Whether a SWE or SRE picks up the phone would simply be a matter of which week you happened to call. Alternatively, if you continued with a traditional tiered escalation format, then an "escalation" to SWE isn't so much a call for a subject matter expert but, instead, for an additional collaborator to help track things down in parallel. Most commonly, we see this as a prelude to a war-room.

A second big departure from prior allocations of roles and responsibilities are the type and scope of our contributions to Site Up. At this point we should be directly improving the products we own or partner on through meaningful engineering contributions.

From this point forward it's all about building reliability and scalability into every product we create or partner with SWE on.

Tools SRE, the Partners

A quick disclaimer: not all of my teams have made it to generation four. For those that have, part of the reason they reached this level was because they had freed up a tremendous amount of their time to focus on the future. We looked for opportunities to allow others to leverage our skills without needing to necessarily talk to us. We created or overhauled services as well as core libraries so that others could be more reliable simply by leveraging our code.

As a team we began to embody "Operations is an engineering problem" by providing leverage to the rest of the company. More importantly, we continued to prioritize the work that would make the most impact for the company, and it naturally led us to more rewarding engineering work.

One Team, One Plan, One Set of Priorities

To advance past Partner SREs, you will need to make two foundational improvements to your team. First, you must participate in **unified SRE-SWE planning**. While overall alignment of teams is mentioned in the "Partner" paragraph, a shared pain needs to be felt by both teams. This helps to drive the unified planning phase. It's important to actually produce a single plan that allocates SRE-SWE resources for the projects where they can add the most value.

Second, you must **leverage the plan to create a unified set of priorities**. This should be a single, stack-ranked set of business priorities that both leadership teams have publicly committed to. These should include priorities such as Talent, Site Up, and Site Secure. By creating the plan and priorities, any engineer in any organization will be able to understand not just what they're doing but why and how they fit into it.

Generation 5: The Engineer

This generation functions as a true north for what SRE should be: fully capable engineers that just so happen to prioritize reliability, scalability, and operability. By this point there should be no further references, save organizational structure itself, to "us vs. them" in policy, day-to-day interaction, or planning. This brings us to the defining characteristic of a generation 5 SRE team: every engineer, regardless of title, organizational affiliation, or the specific job functions of their day-to-day role should be able to answer my favorite question with absolute confidence.

"What is your job?"

And the answer? "My job is to help our company win."

References

[1] Site Up: https://www.linkedin.com/pulse/site-up-benjamin-purgason/.

[2] Every day is Monday in Operations: https://www.linkedin.com/pulse/every-day-monday-operations-benjamin-purgason/.

[3] What gets measured gets fixed: https://www.linkedin.com/pulse/what-gets-measured-fixed-benjamin-purgason/.

[4] Attack the problem, not the person: https://www.linkedin.com/pulse/attack-problem-person-benjamin-purgason/.