The Open Organization Guide to Distributed Teamwork

Copyright

Peter Baumgartner's "The importance of trust on distributed teams" was originally published at https://opensource.com/open-organization/17/7/lincoln-loop-trust.

Laura Hilliger's "How to run an online community meeting (in 11 steps)" was originally published at https://opensource.com/open-organization/16/1/community-calls-will-increase-participation-your-open-organization.

Guy Martin's "It's not the tool: Building a culture of transparency through company-wide chat" was published at https://opensource.com/open-organization/17/12/chat-platform-default-to-open.

Alexis Monville's "Understanding self-organization and management" was originally published at https://opensource.com/open-organization/18/8/self-organizing-team-management.

Chad Whitacre's "Distributed teams benefit when they track issues publicly" was originally published at https://opensource.com/open-organization/17/2/tracking-issues-publicly.

---

1 http://creativecommons.org/licenses/by-sa/4.0/

Colophon

Typeset in DejaVu[2] and Red Hat.[3] Produced with LibreOffice.[4]

2 https://dejavu-fonts.github.io/

3 https://github.com/RedHatOfficial/RedHatFont

4 https://www.libreoffice.org/

Version 0.90

June 2020

Also in the series

Contents

Preface 8
Laura Hilliger & Bryan Behrenshausen

Introduction 10
Ben Cotton

Establishing Trust

The importance of trust on distributed teams 16
Peter Baumgartner

Communicating

It's not the tool: Building a culture of transparency through company-wide chat 21
Guy Martin

Collaborating

Distributed teams benefit when they track issues publicly 29
Chad Whitacre

Becoming a remote-first company by practicing openness 34
Isabel Drost-Fromm

Building community

How to run an online community meeting (in 11 steps) 45
Laura Hilliger

Building a movement from home 51
Chad Sansing

What I learned about working openly after one year on a distributed team 54
Anupama Jha

Leading and managing

Could your team be managing itself? 61
Alexis Monville

About the contributors 65

Appendix

The Open Organization Definition 68
The Open Organization Ambassadors

Learn more

Additional resources 70

Get involved 71

Preface

Laura Hilliger & Bryan Behrenshausen

We're all working in unprecedented times. For months, the novel coronavirus SARS-CoV-2 has swept steadily across the globe. Concerns over the disease it causes, COVID-19, have spurred shifts—both subtle and seismic—in seemingly all facets of everyday life. Phrases like "contact tracing," "social distancing," and "stay at home order" are common parlance. And organizations everywhere are grappling with the unavoidable consequences of a world that increasingly discourages side-by-side collaborations.

In a way, it's a reckoning. Long-held assumptions—about the necessity of co-located work arrangements, the irreplaceable benefits of shared proximity, and the primacy of synchronous interactions—no longer seem so indisputable. Organizations are experimenting every day with new strategies and tactics for achieving their mission when their people shouldn't be sitting within six feet of one another.

Like most open source projects, the Open Organization community was working remotely long before COVID-19 became a global pandemic. Now, however, we realize organizations everywhere might benefit from some of the same principles and practices that open source communities like ours have embraced for decades as they've successfully developed critical technological and social innovations across vast distances. Hence this book.

But be aware: *The Open Organization Guide to Distributed Teamwork* is not about the technology that makes remote work possible. It's about the nuances of human collaboration in an asynchronous, digital-first world.

That's why we actually prefer the term "distributed teamwork" to describe the challenge we're all facing today. "Remote work" sounds like a way to describe someone working from the far reaches of the planet, perhaps isolated in a bunker somewhere. It seems to describe a situation that's both temporary and detached from lived reality, one of *constant connectedness* that now permeates our working lives. We may be physically distant from one another, but we aren't psychologically remote. In fact, we need to be connected in a way that's much more difficult than it would be if we were sharing an office.

Indeed, to work successfully in virtual environments, we have to pay special attention to the humans on the other side of the screen. We don't just need to "be connected"; we need to be *building relationships* and *creating trust* in new and challenging ways. And without a doubt, we need to balance constant connection with more intentional inclusivity.

This book will help readers understand how building teams in distributed environments is inherently different. It will provide tips and tricks, resources, and lessons, all couched in the five principles that make open organizations so successful: transparency, inclusivity, collaboration, community, and adaptability. We hope it will help you empower you and your colleagues to build processes and cultivate cultures that help everyone in the organization to not only cope but *thrive*—no matter how close, or how far apart, they happen to be.

June 2020

Introduction

Ben Cotton

Why might an organization be interested in distributed work? Listing individual reasons could fill this entire guide. Most organizations that choose a distributed work model don't do so for idealistic reasons; they do it because they see it as the best way to meet some organizational need. Maybe it's to save money on the cost of renting, furnishing, and operating office space. Maybe it's to improve hiring, whether by making relocation less expensive or becoming more attractive to members who can't or won't relocate. Maybe it's to have greater resiliency in case of natural disaster, civil unrest, or any of the other calamities in your property insurance policy's fine print. Whatever the reason, deciding to operate in a distributed manner involves a deliberate choice to "work differently."

But what if history had unfolded differently and all the events that led to co-located work becoming the "default" were reversed? Just as organizations can "default to open," they could also "default to distributed."

Sure, some activities don't lend themselves ideally to distributed work. Your neighborhood soup kitchen or hair salon may be shining beacons of openness, but the work that occurs there requires physical co-presence. Fire fighters and surgeons (with some inspiring technological exceptions) can't do their jobs from across the country.

But why not make *those* the exceptional cases?

Let's start with the assumption that our work will be distributed—and then make *centralization* the deliberate choice. We

might soon discover that distributed work is the rule, not the exception.

Why distributed work?

After all, "distributed workers" aren't necessarily working from their homes or their favorite coffee shops. They're not necessarily working from the remote reaches of the Antarctic. Maybe your entire team is in your organization's offices—just not the same office. Or maybe they *are* all in the same office—but they're separated by 14 floors, and gathering for a co-located meeting might take as long as the meeting itself.

A company with tens of thousands employees—even if they're all on the same sprawling campus—are probably not doing most of their work face-to-face. In a previous role, for example I worked for a company where a large percentage of the employees —and almost everyone I interacted with there—spent their days in a corporate headquarters with more than 100 individual buildings. It was basically a city unto itself. I worked from home, so when I made trips to headquarters, I made a concerted effort to attend meetings in person. That's when I noticed I was spending a quarter of my day simply traveling between buildings. It was good for my step count, but it wasn't something that would be efficient if done on a regular basis. Even with everyone in the same approximate physical location, I realized *we were a distributed company*. We just never acknowledged that.

Wherever your organization's members are, distributed work means meeting them there. If you're not already convinced that defaulting to distributed is the right choice, consider the impact it has. Distributed work enables your organization to be more inclusive.[5] By freeing team members from geographic constraints, you show them they're valuable—not just that they're the best you could find within a reasonable commute distance. So if we're al-

5 https://opensource.com/open-organization/19/1/remote-work-inclusivity

ready doing *de facto* distributed work, why could the remote employee experience not be better?

Why this book?

The idea that organizations are already reliant on distributed work processes and practice—even if they aren't consciously acknowledging it—is partly of why we wrote this book. Of course, we also recognize the organization that truly isn't distributed but wants (or needs) to be. And we acknowledge the organization that has embraced distributed work but wants to be better at it. Since distributed work has become an undeniable fact of modern life, your organization is going to fall into at least *one* of these categories.

Distributed work is not without its challenges, of course. That's the *primary* reason we wrote this book, which collects lessons from Open Organization community contributors across the world and in a variety of industries. We wrote (and continue to write) the book as a distributed work project, refining our ideas as we put them into practice. As with so many things in life, it is not the Single Right Answer™. Instead, it's a guide to help you find the answer that works best for *your* organization.

We begin with a section on establishing trust. Trust is important in *any* organizational context, but it's even more critical in distributed organizations where standing at someone's desk and checking on them is rather difficult (if not impossible). Trust—between teammates, between individual contributors and their managers, and between leaders across the distributed organization—is paramount to success in a distributed context. Without trust, the rest of what's in this book doesn't matter.

From there, we move into the more functional aspects of distributed work: communicating and collaborating. Before you can work together to refine ideas, you must be able to share them. This involves not only the *technology* you use for communicating and collaborating but also *the social norms* for *how* you use that technology.

In the fourth section, we build the community for which the first three sections lay a foundation. Two aspects of community are important to distributed organizations. The first is the organization's ability to establish a sense of community among distributed teammates. This includes growing the bonds between people in your organization—helping them connect when they don't get to meet in physical space—and creating a collective sense of purpose. Building the community in this way means you are taking the existing members and helping them be more effective. The second is *scaling* that community by adding new members and weaving them into the fabric of your organization. Building the community in this way means you have more resources available, potentially in new areas that you weren't connected to before.

Lastly, we cover topics related to leading and managing distributed work teams. Just as "traditional" management practices don't always fit an open organization model, distributed work requires a change in management approaches.

We've ordered these sections intentionally, as we've designed this book to reflect the journey that organizations undertake as they embrace distributed work. Seasoned practitioners will recognize that journey as one that follows Tuckman's stages of group development.[6] Bruce Tuckman identified four inevitable and universal stages of group development: forming, storming, norming, and performing. Forming—how teams come to be in the first place—is out of the scope of this book (if you're reading this book, then we assume you've already formed your team). For distributed teams, building trust is integral to the "storming" stage of group development. Despite its name, storming doesn't require contentious arguments, but it *is* the stage where trust gets built. Sections on communication and collaboration speak to elements of the "norming" stage. As the name indicates, this is where the team establishes and accepts norms. And the last two sections reflect activities that occur in Tuckman's "performing" stage. This

6 https://en.wikipedia.org/wiki/Tuckman%27s_stages_of_group_development

is where where the organization truly begins reaping the benefits of remote work.

A work in progress

This book is a work in progress; what you're reading is only the first version. In the spirit of open source, we chose to release early and release often. As additional chapters are ready, we'll publish subsequent releases. And of course, those future releases will also include "bug fixes"—typos and style corrections, yes, but also refinements to ideas. We'll continue to learn from our experiences, each other, and you.

Your contributions will make this book an even more valuable resource to future readers. We welcome your comments and suggestions. Let us know what's missing. What has worked, and what hasn't worked as your organization learns to embrace distributed work? Development is happening on GitHub,[7] so you can see the work as it occurs and provide us with your feedback—or, even better, your own contribution.

7 https://github.com/open-organization/open-org-distributed-work-guide

Establishing Trust

The importance of trust on distributed teams

Peter Baumgartner

Lincoln Loop is an open organization in many ways. We're distributed across 7 time zones.[8] We have no central headquarters. All members of our core team can see all our financials[9] (literally every penny earned or spent) and choose their own salaries.[10] We have an open vacation policy and let people set their own work schedules.

When I tell people how Lincoln Loop operates, the response is typically shock and disbelief followed by a long list of reasons their companies could never operate this way. I often hear things like:

- "We'd go broke if we let people choose their salaries!"
- "Nobody would ever come to work if we let them choose their own schedules!"
- "Employees would take advantage of an open vacation policy!"

What I *really* hear when business owners tell me this is: "I don't trust my employees."

But that attitude is a direct result of outdated command-and-control-style of management. And its effects on creativity and productivity are both negative and profound. At Lincoln Loop, we've

---

8 https://lincolnloop.com/blog/2012/aug/20/distributed-workplace/

9 https://lincolnloop.com/blog/open-book-finances/

10 https://lincolnloop.com/blog/lincoln-loop-everyone-sets-their-own-salary/

learned how placing a high value on organizational trust makes for a happier, more productive workplace.

Productivity drains

Managers who don't trust employees frequently use rules and regulations to box those employees in—the idea being, "with enough rules, there won't be any leeway for employees to make mistakes or cause problems." They try to regulate their way to efficiency.

But when you try to control people with a "my way or the highway" approach, you stifle creativity and often productivity. People aren't all round pegs managers can shove in the round holes they've created.

Here's an easy example of this in practice. A company policy may dictate that people must be in their offices from 8 a.m. to 5 p.m. every work day. If you ask workers, though, many will tell you the office is where they are *least* productive. What's more important to the company, productivity or having warm bodies in the office for a specified period of time? The rule doesn't support the purpose of the company.

Dee Hock, the innovative founder of Visa, said it well:

> To the degree that you hold purpose and principles in common among you, you can dispense with command and control. People will know how to behave in accordance with them, and they'll do it in thousands of unimaginable, creative ways.[11]

11 https://books.google.com/books?id=VWOPCwAAQBAJ&lpg=PT98&dq=dee%20hock%20%22dispense%20with%20command%20and%20control%22&pg=PT98#v=onepage&q=dee%20hock%20%22dispense%20with%20command%20and%20control%22&f=false

Morale killers

Excessive workplace rules are so commonplace that we've become complacent about them. We don't look at the emotional effect they're having on people.

Consider a slightly different scenario but with similar rules. What if somebody invited you to a dinner party, but made a point to tell you what clothes to wear and how to comb your hair and brush your teeth before coming over? The implication here is that you're incapable of making good decisions on your own. You might treat a child like this—but not a grown adult. So why are we complacent with it in the workplace?

When you treat your employees like children, don't be surprised if you also have issues with morale—people only doing the bare minimum and counting down the minutes until the end of the day.

Reversing the trend

The bottom line, though, is that distrust breeds more distrust. When management communicates a lack of trust to employees, those employees will reciprocate with a lack of trust for management. When this relationship becomes adversarial, the organization breaks down. Retaining employees becomes difficult and the workplace becomes toxic.

At Lincoln Loop, we work with trustworthy people. We give them tasks and *trust* them to complete them however they see fit. If their work requires collaboration, we assume they'll ensure that happens. If meeting a deadline is necessary, we likewise assume they'll meet it or raise a flag if they can't.

Where, when, or how people complete their tasks are rarely of importance to us. Instead, we focus our energy on making sure people understand the *big picture*: how their task fits into the company's mission and goals.

Our trust in our team lets us be very light on rules without devolving into anarchy. We share an explicit set of core values, which everyone uses to guide their decisions and daily work. When

you work with good people and you you trust them to do their jobs, guess what? They do!

And not only that, but they're much happier doing them. Instead of being a nameless cog in a machine, they feel empowered to make decisions that are important to the business. That trust breeds a sense of community and teamwork far more effectively than any motivational speaker or company holiday party ever could.

At Lincoln Loop, our emphasis on trust has led to results that make me extremely proud. Most members of our core team have been with the company for more than eight years (in an industry where even *two years* sounds like the norm). Major competitors have pursued many of them, presumably offering more money and better tangible benefits. From what I hear, however, those interviews typically last until they ask about open practices and policies —like remote work and flexible hours. Our open organization always beats what they have to offer.

Communicating

It's not the tool: Building a culture of transparency through company-wide chat

Guy Martin

Collaboration and information silos are a reality in most organizations today. People tend to regard them as huge barriers to innovation and organizational efficiency. They're also a favorite target for solutions from software tool vendors of all types.

Tools by themselves, however, are seldom (if ever), the answer to a problem like organizational silos. The reason for this is simple: Silos are made of people, and human dynamics are key drivers for the existence of silos in the first place.

So what *is* the answer?

Successful communities are the key to breaking down silos. Tools play an important role in the process, but if you don't build successful communities around those tools, then you'll face an uphill battle with limited chances for success. Tools *enable* communities; they do not build them. This takes a thoughtful approach—one that looks at culture first, process second, and tools last.

However, this is a challenge because, in most cases, this is not the way the process works in most businesses. Too many companies begin their journey to fix silos by thinking about tools first and considering metrics that don't evaluate the right factors for success. Too often, people choose tools for purely cost-based, compliance-based, or effort-based reasons—instead of factoring in the needs and desires of the user base. But subjective measures like "customer/user delight" are a real factor for these internal tools,

and can make or break the success of both the tool adoption and the goal of increased collaboration.

It's critical to understand the best technical tool (or what the business may consider the most cost-effective) is not always the solution that drives community, transparency, and collaboration forward. There is a reason that "Shadow IT"—users choosing their own tool solution, building community and critical mass around them—exists and is so effective: People who choose their own tools are more likely to stay engaged and bring others with them, breaking down silos organically.

This is a story of how Autodesk ended up adopting Slack at enterprise scale to help solve our transparency and silo problems. Interestingly, Slack wasn't (and isn't) an IT-supported application at Autodesk. It's an enterprise solution that was adopted, built, and is still run by a group of passionate volunteers who are committed to a "default to open" paradigm.

Utilizing Slack made transparency happen for us.

Chat-tastrophe

First, some perspective: My former role at Autodesk involved running our Open@ADSK initiative. I was originally hired to drive our open source strategy, but we quickly expanded my role to include driving open source best practices for internal development, and transforming how we collaborate internally as an organization. This last piece is where we pick up our story of Slack adoption in the company.

But before we even begin to talk about our journey with Slack, let's address why lack of transparency and openness was a challenge for us. What is it that makes transparency such a desirable quality in organizations, and what was I facing when I started at Autodesk?

Every company says they want "better collaboration." Autodesk is a 35-year-old software company that has been immensely successful at selling desktop "shrink-wrapped" software to several industries, including architecture, engineering, construction, man-

ufacturing, and entertainment. But no successful company rests on its laurels, and Autodesk leadership recognized that a move to Cloud-based solutions for our products was key to the future growth of the company, including opening up new markets through product combinations that required Cloud computing and deep product integrations.

The challenge in making this move was far more than just technical or architectural—it was rooted in the DNA of the company, in everything from how we were organized to how we integrated our products. The basic format of integration in our desktop products was file import/export. While this is undoubtedly important, it led to a culture of highly-specialized teams working in an environment that's more siloed than we'd like and not sharing information (or code). Prior to the move to a cloud-based approach, this wasn't as a much of a problem—but, in an environment that requires organizations to behave more like open source projects do, transparency, openness, and collaboration go from "nice-to-have" to "business critical."

Like many companies our size, Autodesk has had many different collaboration solutions through the years, some of them commercial, and many of them home-grown. However, none of them effectively solved the many-to-many real-time collaboration challenge. Some reasons for this were technical, but many of them were cultural.

When someone first tasked me with trying to find a solution for this, I relied on a philosophy I'd formed through challenging experiences in my career: "Culture first, tools last." This is still a challenge for engineering folks like myself. We want to jump immediately to tools as the solution to any problem. However, it's critical to evaluate a company's ethos (culture), as well as existing processes to determine what kinds of tools might be a good fit. Unfortunately, I've seen too many cases where leaders have dictated a tool choice from above, based on the factors discussed earlier. I needed a different approach that relied more on fitting a

tool into the culture we wanted to become, not the other way around.

What I found at Autodesk were several small camps of people using tools like HipChat, IRC, Microsoft Lync, and others, to try to meet their needs. However, the most interesting thing I found was *85 separate instances of Slack in the company!*

Eureka! I'd stumbled onto a viral success (one enabled by Slack's ability to easily spin up "free" instances). I'd also landed squarely in what I like to call "silo-land."

All of those instances were not talking to each other—so, effectively, we'd created isolated islands of information that, while useful to those in them, couldn't transform the way we operated as an enterprise. Essentially, our existing organizational culture was recreated in digital format in these separate Slack systems. Our organization housed a mix of these small, free instances, as well as multiple paid instances, which also meant we were not taking advantage of a common billing arrangement.

My first (open source) thought was: "Hey, why aren't we using IRC, or some other open source tool, for this?" I quickly realized that didn't matter, as our open source engineers weren't the only people using Slack. People from all areas of the company—even senior leadership—were adopting Slack in droves, and, in some cases, convincing their management to pay for it!

My second (engineering) thought was: "Oh, this is simple. We just collapse all 85 of those instances into a single cohesive Slack instance." What soon became obvious was that was the easy part of the solution. Much harder was the work of cajoling, convincing, and moving people to a single, transparent instance. Building in the "guard rails" to enable a closed source tool to provide this transparency was key. These guard rails came in the form of processes, guidelines, and community norms that were the hardest part of this transformation.

The real work begins

As I began to slowly help users migrate to the common instance (paying for it was also a challenge, but a topic for another day), I discovered a dedicated group of power users who were helping each other in the #adsk-slack-help channel on our new common instance of Slack. These power users were, in effect, building the roots of our transparency and community through their efforts.

The open source community manager in me quickly realized these users were the path to successfully scaling Slack at Autodesk. I enlisted five of them to help me, and, together we set about fabricating the community structure for the tool's rollout.

Here I should note the distinction between a community structure/governance model and traditional IT policies: With the exception of security and data privacy/legal policies, volunteer admins and user community members completely define and govern our Slack instance. One of the keys to our success with Slack (currently approximately 9,100 users and roughly 4,300 public channels) was how we engaged and involved our users in building these governance structures. Things like channel naming conventions and our growing list of frequently asked questions were organic and have continued in that same vein. Our community members feel like their voices are heard (even if some disagree), and that they have been a part of the success of our deployment of Slack.

We did, however, learn an important lesson about transparency and company culture along the way.

It's not the tool

When we first launched our main Slack instance, we left the ability for anyone to make a channel private turned on. After about three months of usage, we saw a clear trend: More people were creating *private channels* (and messages) than they were *public channels* (the ratio was about two to one, private versus public). Since our effort to merge 85 Slack instances was intended to in-

crease participation and transparency, we quickly adjusted our policy and turned off this feature for regular users. We instead implemented a policy of review by the admin team, with clear criteria (finance, legal, personnel discussions among the reasons) defined for private channels.

This was probably the only time in this entire process that I regretted something.

We took an amazing amount of flak for this decision because we were dealing with a corporate culture that was used to working in independent units that had minimal interaction with each other. Our defining moment of clarity (and the tipping point where things started to get better) occurred in an all-hands meeting when one of our senior executives asked me to address a question about Slack. I stood up to answer the question, and said (paraphrased from memory): "It's not about the tool. I could give you all the best, gold-plated collaboration platform in existence, but we aren't going to be successful if we don't change our approach to collaboration and learn to *default to open*."

I didn't think anything more about that statement—until that senior executive starting using the phrase "default to open" in his slide decks, in his staff meetings, and with everyone he met. That one moment has defined what we have been trying to do with Slack: The tool isn't the sole reason we've been successful; it's the approach that we've taken around building a self-sustaining community that not only wants to use this tool, but craves the ability it gives them to work easily across the enterprise.

What we learned

I say all the time that this could have happened with other, similar tools (Hipchat, IRC, etc.), but it works in this case specifically because we chose an approach of supporting a solution that the user community adopted for their needs, not strictly what the company may have chosen if the decision was coming from the top of the organizational chart. We put a lot of work into making it an acceptable solution (from the perspectives of security, legal, fi-

nance, etc.) for the company, but, ultimately, our success has come from the fact that we built this rollout (and continue to run the tool) as a community, not as a traditional corporate IT system.

The most important lesson I learned through all of this is that transparency and community are evolutionary, not revolutionary. You have to understand where your culture is, where you want it to go, and utilize the lever points that the community is adopting itself to make sustained and significant progress. There is a fine balance point between an anarchy, and a thriving community, and we've tried to model our approach on the successful practices of today's thriving open source communities.

Communities are personal. Tools come and go, but keeping your community at the forefront of your push to transparency is the key to success.

Collaborating

Distributed teams benefit when they track issues publicly

Chad Whitacre

A public issue tracker is a vital communication tool for an open organization, because there's no better way to be transparent and inclusive than to conduct your work in public channels. So let's explore some best practices for using an issue tracker in an open organization.

Before we start, though, let's define what we mean by "issue tracker." Simply put, an issue tracker is a *shared to-do list*. Think of scribbling a quick list of errands to run: buy bread, mail package, drop off library books. As you drive around town, it feels good to cross each item off your list. Now scale that up to the work you have to do in your organization, and add in a healthy dose of software-enabled collaboration. You've got an issue tracker!

Whether you use GitHub, or another option, such as Jira, GitLab, or Trello, an issue tracker is the right tool for the task of coordinating with your colleagues. It is also crucial for converting outsiders into colleagues, one of the hallmarks of an open organization. How does that work? I'm glad you asked!

Best practices for using an issue tracker

The following best practices for using a public issue tracker to convert outsiders into colleagues are based on our experience at Gratipay over the past five years. We help companies and others pay for open source, and we love collaborating with our community using our issue trackers. Here's what we've learned.

0. Prioritize privacy

It may seem like an odd place to start, talking about privacy in a post about public issue trackers. But we must remember that openness is not an end in itself, and that any genuine and true openness is only ever built on a solid foundation of safety and consent.[12] Never post information publicly that customers or other third parties have given you privately, unless you explicitly ask them and they explicitly agree to it. Adopt a policy and train your people. Okay! Now that we're clear on that, let's proceed.

1. Default to deciding in public

If you make decisions in private, you're losing out on the benefits of running an open organization, such as surfacing diverse ideas, recruiting motivated talent, and realizing greater accountability. Even if your full-time employees are the only ones using your public issue tracker at first, do it anyway. Avoid the temptation to treat your public issue tracker as a second-class citizen. If you have a conversation in the office, post a summary on the public issue tracker, and give your community time to respond before finalizing the decision. This is the first step towards using your issue tracker to unlock the power of open for your organization: if it's not in the issue tracker, it didn't happen!

2. Cross-link to other tools

It's no secret that many of us love IRC and Slack. Or perhaps your organization already uses Trello, but you'd like to start using GitHub as well. No problem! It's easy to drop a link to a Trello card in a GitHub issue, and vice versa. Cross-linking ensures that an outsider who stumbles upon one or the other will be able to discover the additional context they need to fully understand the issue. For chat services, you may need to configure public logging in order to maintain the connection (privacy note: when you do so, be sure to advertise the fact in your channel description). That

12 https://opensource.com/open-organization/16/9/openness-means-to-what-end

said, you should treat conversations in private Slack or other private channels just as if they were face-to-face conversations in the office. In other words, be sure to summarize the conversation on the public issue tracker. See above: whether offline or online, if it's not in the issue tracker, it didn't happen!

3. Drive conversations to your tracker

Social media is great for getting lots of feedback quickly, and especially for discovering problems, but it's not the place to solve them. Issue trackers make room for deeper conversations and root-cause analysis. More importantly, they are optimized for getting stuff done rather than for infinite scrolling. Clicking that "Close" button when an issue is resolved feels really good! Now that you have a public issue tracker as your primary venue for work, you can start inviting outsiders that engage with you on social media to pursue the conversation further in the tracker. Something as simple as, "Thanks for the feedback! Sounds similar to (link to public issue)?" can go a long way towards communicating to outsiders that your organization has nothing to hide, and welcomes their engagement.

4. Set up a "meta" tracker

Starting out, it's natural that your issue tracker will be focused on your product. When you're ready to take open to the next level, consider setting up an issue tracker about your organization itself. At Gratipay, we're willing to discuss just about any aspect of our organization, from our budget[13] to our legal structure[14] to our name,[15] in a public issue tracker we call "Inside Gratipay." Yes, this can get a little chaotic at times—renaming the organization was a particularly fierce bikeshed!—but for us the benefits in terms of community engagement are worth it.

13 https://github.com/gratipay/inside.gratipay.com/issues/928

14 https://github.com/gratipay/inside.gratipay.com/issues/72

15 https://github.com/gratipay/inside.gratipay.com/issues/73

5. Use your meta tracker for onboarding

Once you have a meta issue tracker, a new onboarding process suggests itself: invite potential colleagues to create their own onboarding ticket. If they've never used your particular issue tracker before, it will be a great chance for them to learn. Registering an account and filing an issue should be pretty easy (if it's not, consider switching tools!). This will create an early success event for your new colleague, as well as the beginnings of a sense of shared ownership and having a place within the organization. There are no dumb questions, of course, but this is especially true in someone's onboarding ticket. This is your new colleague's place to ask any and all questions as they familiarize themselves with how your organization works. Of course, you'll want to make sure that you respond quickly to their questions, to keep them engaged and help them integrate into your organization. This is also a great way to document the access permissions to various systems that you end up granting to this person. Crucially, this can start to happen before they're even hired.[16]

6. Radar projects

Most issue trackers include some way to organize and prioritize tasks. GitHub, for example, has milestones and projects. These are generally intended to align work priorities across members of your organization. At Gratipay, we've found it helpful to also use these tools to allow collaborators to own and organize their individual work priorities. We've found this to offer a different value than assigning issues to particular individuals (another facility issue trackers generally provide). I may care about an issue that someone else is actively working on, or I may be potentially interested in starting something but happy to let someone else claim it first. Having my own project space to organize my view of the organization's work is a powerful way to communicate with my colleagues about "what's on my radar."

16 https://opensource.com/open-organization/16/5/employees-let-them-hire-themselves

7. Use bots to automate tasks

Eventually, you may find that certain tasks keep popping up again and again. That's a sign that automation can streamline your workflow. At Gratipay, we built a bot to help us with certain recurring tasks. Admittedly, this is a somewhat advanced use case. If you reach this point, you will be far along in using a public issue tracker to open up your organization!

Those are some of the practices that we've found most helpful at Gratipay in using our issue tracker to "engage participative communities both inside and out," as Jim Whitehurst puts it in *The Open Organization*.

That said, we're always learning.

Becoming a remote-first company by practicing openness

Isabel Drost-Fromm

As organizations grow, they often enter a phase in which internal teams become more independent from each other. But when they do, they also begin repeating work that has been done elsewhere. Knowledge no longer flows easily from one team to another. Innovation suffers. Enabling transparent communication is one way to help these teams utilize resources already present inside an enterprise and avoid duplicate work. Combining InnerSource and open organization principles can furnish the tools necessary for supporting this transparent communication. I started learning this lesson three years ago when my own employer—Europace, a mid-sized company in Germany—embarked on a journey to adopt InnerSource as a development principle. Three years ago, most of our employees were still located in one office. Remote work was possible, but remote-only colleagues were the exception rather than the norm.

But throughout those three years, we successfully set the organizational foundation for a seamless move to remote-first work culture. And we did so just in time, as Europace became an all-remote organization during the COVID-19 pandemic.

In this chapter, I'll discuss the ways that working openly prepared our organization for a successful shift to full-time remote work. I'll explain how adopting and incorporating an increasing number of InnerSource patterns into our daily work helped us reduce the effects of organizational silos between units. I'll also explain how colleagues were able to move to asynchronous com-

munication patterns, allowing senior contributors to adopt new work schedules while still being able to participate and add their knowledge in day-to-day software development. I'll describe how we were able to clarify role definitions to help re-shape teams that had grown substantially, so we could map a mixed monolithic/micro-service architecture more easily to our team's structure. And I will recount how the new style of working helped us embrace open decision-making practices.

Open practices at Europace

InnerSource, according to InnerSource Commons, "takes the lessons learned from developing open source software and applies them to the way companies develop software internally." It "can be a great tool to help break down silos, encourage internal collaboration, accelerate new engineer on-boarding, and identify opportunities to contribute software back to the open source world." In order to achieve these goals, InnerSource works with concepts that are fairly similar to the ways open source projects operate—that is, teams share projects internally and collaborate on them like open source communities do.

By embracing InnerSource, teams do more than share code internally, however. They also adopt communication and decision making practices typical of open source communities. And like those communities, they externalize and visualize their planning processes, making them visible across the organization. Most importantly, all project communication related to an InnerSource project happens in writing, so it's visible to all company members in an archived medium that anyone can search and link to. The goal is to make work transparent so others can join the project. This degree of organizational transparency helps bring teams closer together—and, potentially, remove silos (or avoid them altogether).

At Europace, we combined InnerSource practices with some tools and principles from the Open Organization community (which also stresses the importance of transparency for working openly).

For example, we adopted a simplified version of the Open Decision Framework to develop a process for company-wide decision making. Europace AG offers the leading web-based Europace platform for the sale of mortgages and related financial and insurance products. The platform links products and services of financial distributors, banks, building societies as well as insurance companies.

The Europace Marketplace has been successfully established in the German financing market since 1999 and offers distributors the best conditions to successfully and independently advise consumers on real estate financing and installment loans.

Back in 2017, the Europace teams faced multiple challenges:

- At a cross-team level, working on a common platform while achieving full team autonomy wasn't possible.
- Former technical contributors moved to leadership roles.
- Cross-team coordination that was easily and informally possible during early stages of organizational growth became infeasible as the organization continued growing.
- Focusing primarily on team autonomy led to a lack of alignment which in turn meant that multiple solutions were developed for similar problems.
- When teams started to grow beyond their initial "pizza-size" structure, communication overhead started to increase.

In order to tackle the issues we were faced with at Europace we initiated an iterative process to adjust collaboration practices. In each iteration, one team selected one of it's most pressing issues to address. We then ran an experiment to fix the issue. We subsequently shared everything we learned with the rest of the company—and with both the public, where applicable (through the public company blog), and the InnerSource Commons, who helped us identify substantial patterns.

As a result, we were able to alleviate many of our pain points.

Dealing with cross-team dependencies

What is so particularly difficult about scaling a team that initially built one common platform?

Ideally, in order to move quickly, teams should operate independently from each other. They should "pull in" people with skills they need—requirements engineering, software development, user experience, design, testing and operations, etc.—as they need them, in order to reduce handover.

In reality, this is an oversimplification to say the least.

Too often, teams tend to overlook the fact that following good engineering practice and re-using pre-existing libraries and services not only creates technical dependencies but also introduces new social and cultural dependencies as well—dependencies, that is, on other teams. And working in an open organization introduces an additional dimension to the issue of dependencies: Even if a development team can move independently from everyone else in a company (and, typically, it cannot), so much of the foundation on which it builds consists of open source software that depends on open source projects producing this software.

Sooner or later, when teams need to make changes that involve those dependencies, they'll face a choice: create workarounds, or wait for another team to fix the issue at hand. Obviously, neither solution is optimal. In our case at Europace, both led to frustration and conflict between teams (especially when it came to prioritization).

But open source communities have shown us a third solution: invest a little time from both teams, and let users implement the changes they need themselves. InnerSource projects encourage the same behaviors. As a result, teams are much more fluent compared to traditional organizations.

At Europace, then, we made several attempts to come up with a software architecture modular enough to allow teams to operate independently. Thinking in roles and patterns common to InnerSource projects helped us alleviate this problem: * The team

providing a dependency is treated as the "host team, often made up of what InnerSource practitioners call"trusted committers." They not only set strategic and technical direction but are also responsible for mentoring newcomers, setting a tone for the project, and enabling new contributors. * Members of the team using a component as a dependency can act as contributors to it, making changes to that component in coordination with and with support from trusted committers.

In this way, then, contributors can modify components on which they rely for their own work. They won't be able to make these modifications as quickly as trusted committers, but with support from trusted committers they are able to work on the modification earlier (and likely finish it earlier, too). Over time their knowledge of the component on which they depend improves—and with it, so does "ramp-up" time for making modifications. This is not unlike what happens when teams start contributing modifications to the open source projects they depend upon. The first fix may take a while, but over time developers become more and more familiar with the code base (as well as the open source project's workflows for making modifications) and can enact that workflow with increasing speed. This similarity is by design. The goal when creating the InnerSource Commons was to raise awareness of how participants in open source projects collaborate in order to lower the bar for participation for paid developers to improve the components they rely on on a daily basis.

One precondition for success with this model is that all (important) communication around projects must be visible to everyone. In addition to using a version control system the contents of which are visible to all teams, this also means tracking important decision making processes in a way that is visible to everyone. Simply telling everyone to "just use GitHub," for example, doesn't help entirely here; even when moving code to GitHub Enterprise Cloud, developers must still be familiar with reading and reviewing changes in patch format. They need to understand how to formulate a pull request in a way that's easy to understand. And

when coming from a co-located, "everyone is in the office and communicates face-to-face" environment, they'll need to learn to provide their feedback in writing—a medium in which many of the visual cues (like gestures) are not available. While seemingly simple, making expectations explicit—for instance, when it comes to review turnaround times—does help eliminate frustration.

Changing team member roles

Every team reaches a point where some of its contributors change teams, accept formal leadership positions, or move to other roles that pivot their focus away from technical contributions. Often, even after taking extensive handover measures, a development team will experience a loss of knowledge after someone has moved on. Adopting an InnerSource model like the one above has a critical side effect: helping teams deal with this problem.

As all communication happens in writing by default, it's recorded and visible to everyone—including those working on a different schedule. When questions arise, answering them becomes much easier when the team can pull in someone else without having to recall and repeat the entire conversation that has unfolded up until that point.

How does that help with colleagues changing roles?

Conventional teams often rely on techniques like "pair programming" or "mob programming" to help junior engineers tap into knowledge from other team members. Techniques like those are great tools to facilitate knowledge sharing and collaborative work. But they have a significant disadvantage that makes them less attractive when practiced remotely: all the communication that occurs during pair or mob programming sessions ephemeral—so anyone joining later doesn't have a chance to follow the design process or the arguments and decisions that lead to a final solution.

Additionally, these techniques really require everyone to be in the same room (or at least the same virtual room). When team

members move to different roles, their schedules often change substantially. Pulling them into pair sessions becomes much more difficult (as they're now working on what Paul Graham calls a "manager's schedule," not a "maker's schedule"). Relying on asynchronous communication can help alleviate this problem. This way, people working on a manager's schedule can still participate and support development, even when they're not participating synchronously in programming sessions. The focus on sharing progress early means that developers can work on solutions and implementations, share them in a "work in progress" format with others, and receive early feedback about whether they're on the right track.

And as a side-effect of relying on asynchronous conversations, communication also becomes more transparent across time zones.

Making decisions asynchronously

If we look closely at the way InnerSource thinking suggests dealing with cross-team dependencies and the way it helps team members move out of pure development roles—and if we examine some case studies—one ingredient for both successful InnerSource and successful open source becomes clear: A preference for communication that allows for asynchronous participation helps make the project accessible to more contributors.

Customarily, asynchronous communication stresses written over spoken communication—and collecting written messages in a permanent, searchable archive in which messages can be referenced through stable links. Conventional open source projects prefer archived mailing lists for this task (often in combination with a public issue tracker for more structured conversations). We can apply this observation to settings outside the scope of software development, too.

At The Apache Software Foundation, for example, even board-level decision-making relies heavily on asynchronous communication. Similarly at Europace, we realized that we can use

what we learned in software engineering projects and apply that knowledge to cases like coordinating architectural decision-making across multiple departments. Applying language and definitions from InnerSource projects at that level seemed like a hack, so instead we adopted a simplified version of the Open Decision Framework for those cases.

Setting cross-team standards and best practices in the open

In small organizations (even those with more than one team) system architecture tends to be fairly coherent. Cross-cutting concerns tend to get addressed when team members choose a technology to solve a problem. API design tends to be coherent, as everyone involved knows the general design patterns in use.

As organizations grow, however, decisions that affect more than a handful of teams often become cumbersome. For example:

- Teams have different contexts and preferences, and often those preferences aren't well understood beyond team boundaries.
- Trying to get everyone around one table in meetings or workshops becomes more challenging, as more people are potentially involved.

At Europace, we've established a four-step, open process for solving architectural questions across departmental boundaries (though another option for solving these issues is to make use of the ideas in the Open Decision Framework). Our idea was to build a transparent process, one that's open to everyone and focused on those who have a stake in the decision at hand. Here's what we came up with:

1. First, a problem in need of a common, cross unit/ cross team solution should be described in writing in an issue tracker. In that issue, everyone is invited to participate. The goal is to reach a shared understanding of the problem and sketch possible solutions together.
2. Assuming that participants were able to arrive at an agreeable solution during that discussion, the team

should then write a formal proposal and post it as a pull request. The goal here is to let people participate in the solution design process. Some key aspects of this phase are:

    1. Explicitly inviting participation from potential stakeholders who are not yet aware of the discussion
    2. Having all relevant communication tracked, so people who join the discussion later can follow what was already discussed
    3. In the case of competing alternative solutions, documenting why those alternatives were not implemented or pursued
    4. Describing what the solution should cover—its advantages and implications, and a description of how the solution addresses the original problem (not to mention the limits of the solution)
    5. To make communication clearer (and in cases where the goal is to create a cross-team standard for solving a specific technical problem), using specific words and phrases endorsed by the Internet Engineering Task Force when specifying characteristics

3. If the proposal meets no veto (and any veto must be accompanied by a detailed explanation), and if at least two senior contributors explicitly vote in favor of the proposed solution, it can be counted as accepted, is merged, and gets promoted across all teams affected going forward. The intention for those senior contributors is to vote independently of their local context in favor of what, in their perspective, is best for the entire platform (not unlike at The Apache Software Foundation, where committers are expected to act on behalf of the project instead of their individual employer).

This process is advantageous because everyone can participate. It limits the need for meetings; everything relatively uncontroversial can be fleshed out in writing. Only items specifically requiring face-to-face discussions prompt face-to-face meetings. Participants can work on developing solutions on their own time. And people arriving in discussions at a later stage still have a chance of following what happened.

Permanently remote-first

All of the steps above helped our team at Europace normalize asynchronous communication as part of team culture. So in 2020, when the COVID-19 pandemic required all of Europace AG to work remotely, this groundwork was already in place—making it easier for colleagues who wanted to work fully remote to do so. Indeed, the entire organization was able to pivot to remote work in a matter of days.

But even more impressively, introducing these changes to how we work increased transparency of decision-making processes and enabled development to accelerate as teammates remained connected. Having experience with asynchronous communication patterns helped the organization avoid video conference-fatigue. Years of experimentation with InnerSource and open organizational techniques made the switch to a fully remote mode of work much easier—so much so, in fact, that even after Germany's contact restrictions loosen, development across the entire company will remain remote-first.

Building community

How to run an online community meeting (in 11 steps)

Laura Hilliger

Open organizations explicitly invite participation from external communities, because these organizations know their products and programs are world class only if they include a variety of perspectives at all phases of development. Liaising with and assisting those communities is critical. And community calls are my favorite method for interacting with stakeholders both inside and outside an organization. In this chapter, I'll share best practices for community calls and talk a little about how they can spur growth.

What is a community call?

You might think of community calls as office hours for particular themes, or as design sessions for programs. A community call is a meeting, held online, that invites people to gather at a specific time each week or month. They're recurring and open to anyone who wishes to join. A community call is a tool for solving problems, breaking out of individual silos, and finding points of connection between different initiatives or people. Most importantly, a great call serves as a launchpad for communities. Community calls bring people together from all over the world. They serve as a social and cultural touchstone. It's all about connection.

How does a community call develop leadership?

A community call demonstrates transparency and collaboration. Invite others to speak. Facilitate a conversation. Don't give a presentation. A good community call invites people to have ideas,

speak their minds and talk about their own work. Good leaders promote action, build partnerships, motivate, and empower. A community call develops these skills through thoughtful and fun facilitation. Community calls thrive on open practices. Open planning, documentation, and open reflection are essential components.

Running a community call

0. Commit

To plan, execute, and run a successful community call, you'll need to commit a bit of time and energy. In the beginning, you'll need a day to think about and document what you hope to achieve, and you'll need another day to do logistical setup and promotion. If your call is successful, you'll need several hours for each call. You will need an hour to write the agenda each week or month, time to promote, an hour to run the call, another hour to write a reflection, followed by more promotion. A successful community call is a production, but as you get used to it and as your community grows, this will become second nature.

1. Determine your community's identity

If you're planning on running a community call, you need an idea of the community you're trying to build. Of course, your target audience is "everyone," but to gain traction, you need to reach influencers first. Define what you hope to "get out" of the call. This will focus your community.

I've run community calls that focused on using stories to create impact, especially for people who were interested in Greenpeace and the 7 Shifts towards Open.[17] I've co-facilitated barn-raising calls for the Badge Wiki community[18] and educators developing the Web Literacy Map.[19] Community calls are a great

17 https://opensource.com/open-organization/16/1/greenpeace-makes-7-shifts-toward-open

18 https://badge.wiki/wiki/Main_Page

way to jumpstart a community and engage with people around projects and ideas.

2. Record the purpose of your call

You're going to be asking people to join your community call. How will you pitch it to them? Write a few sentences summarizing what your call is about. You can use these sentences to help you promote the call. For example:

> Our community call surfaces real world examples that embody the new Greenpeace Story and the 7 shifts. We want to make this community a network that can engage around all things story.
>
> We hope to celebrate one another, identify gaps in knowledge, and share skills. We serve as a peer group for developing campaign ideas and breathe life into the stories we tell at Greenpeace. We want to build campaigns that sing, dance, dream, laugh, and otherwise grasp the Story and Shifts.
>
> People will be invited to bring what they're working on or need help with. We hope people bring story ideas with the aim of improving campaigns, actions and engagement. We see this as a fast, loose way to collaborate and innovate.

3. Pick a date

Before promoting your call, you need to schedule it! Choose a date and a time when most folks are likely available. Do most members of your community have day jobs? Schedule the call for the early evening. Is your community based primarily in Europe? Plan the call at a time convenient for Europeans. There's no science to picking the best time. Just find a day/time that works for

19 https://foundation.mozilla.org/en/initiatives/web-literacy/

most of your community members. An inclusive technique to scheduling is to run your call three times, once for Europe/Americas, Asia/Oceania and once for Europe/Asia. Just reuse the same agenda and get input from different cultures, regions, languages—these perspective shifts can be amazingly beneficial.

4. Choose your tech

Next, select the particular technologies you'll use to gather everyone together. I recommend using video conferencing software. You can use anything from Zoom to Jitsi, Microsoft Teams to Big Blue Button. Choose something your target community is already using and try it out. If they're all in a Google Community, try Meet. If you communicate via a specific forum or mailing list, use that list to ask people what they use for video conferences.

5. Write the agenda

You don't want to waste people's time, so you need a plan. If your community doesn't feel like the call is valuable, they won't come back. They also won't spread the word, and your community call will have no…community. Balance your agenda. Provide enough presentation so people have something to respond to, but offer enough time for interaction, too, so people feel invited to speak. Use a collaborative note taking device to plan your call. This is the place you can store the logistical information (when the call is, how to connect, etc). It also serves as a living document for note taking and collaboration.

6. Reach out to influencers

To grow your community call, you need to find like-minded people who are interested in the topics. They can help you spread the word. They are also the people most likely to have the skills necessary to collaborate or discuss "the work." For example, the story community call is open to everyone, but I focus promotional efforts on people who understand (or at least know) what "story as a theory of change" means. Generate a list of influencers speaking prominently on your call's theme, then reach out to them individually. Ask them to attend your call. The personal and specific

request is much more powerful than a mass email. This is a critical step!

7. Promote your call

In the two weeks leading up to your call, you'll want to share your agenda widely. Use social media to promote your call. Send personal messages and emails to lists. Ask people to come to your kick-off call. Involve your community members from the beginning. Ask their opinions. Ask for input.

8. Arrive early

Load your agenda and connect to the video or phone conference at least 15 minutes before everyone else is scheduled to join you. You can prepare your opening, make sure the tech is working, and be ready when people start to arrive. Give people some time to connect. I usually "officially" start the call at about five minutes past the scheduled call time. In the meantime, welcome people to the call and explain how to use the collaborative notes document for those who are connecting.

9. Be a superstar moderator

When you're ready to start the call, begin by welcoming your participants and reviewing the meeting's purpose. Explain how to use the notes document. Explain how the call works. Let people know you encourage their participation! Give an overview of the agenda and ask participants to help you take notes. During the call, invite others to speak. If someone is getting off topic, gently refocus the conversation. Listen. I can't stress this enough: *Listen to your community.* Make sure you take notes, too! When your time is up, invite people back by informing them of the next call. And don't forget to say "thank you!"

10. Write a reflection

Here's where the notes come in handy. Documentation is integral to open organizations. A community call can help you make decisions, advance ideas, and assign tasks, but only if you take care to document your conversation and follow up! Write a per-

sonal reflection after each call. Just read the notes and write down what the call experience was like for you. Write your reflections soon after concluding the call, so you remember what you felt, what the conversation covered, and how people interacted. Then, as you begin promoting your next community call, share the new agenda as well as the reflection.

I love community calls. I've been attending them and running them for a long time. Read about my experiences on my blog[20] —or get in touch![21]

20 http://www.laurahilliger.com/?s=community+call

21 http://laurahilliger.com/

Building a movement from home

Chad Sansing

As part of its response to the COVID-19 pandemic, Mozilla hosted a series of "Movement-Building from Home" community calls designed to bring people together for peer-to-peer learning about running online meetings and practicing community care, personal ecology, and community management during and after the immediate crisis.

We wanted to support people new to online facilitation and movement-building. We also wanted to help experienced online activists scale up their work and welcome new community members to their online spaces. And we wanted to hold time and space for community managers and organizers to be together, to acknowledge their challenges, feelings, and stressors, and to celebrate their successes during anxious and uncertain times.

Throughout four weeks of calls, more than 100 live participants joined us to hold that space and share those insights, best practices, questions, and concerns. By engaging with one another and investing in each other, we developed a shared understanding of our work as a powerful example of collaboration for a healthier, more inclusive internet.

Here are the top three lessons we learned from our curious, generous participants.

Prioritize audience participation

Be sure to tilt the balance of each call or meeting in favor of accessible, peer-to-peer interaction over lecture and other kinds of more passive participation. Mix and match approaches like:

- Silent documenting, or sharing thoughts by typing into a shared doc ahead of discussion
- Small-group discussions like Zoom's Breakout Rooms, which help people share the spotlight and bring insights back to the larger group
- Invitations to contribute affirmations like emojis and "+1s" and questions to others' comments instead of—or as well as—more direct responses to prompts and questions

Be responsive

Always invite feedback on your work so you can improve calls and meetings for your participants. You might:

- Establish a feedback ritual at the end of each call, asking for responses to prompts like what worked, what didn't work, what surprised you, and what would you change?
- Link to a survey at the end of each call or in a weekly emailed summary of your events to find out what people want more or less of in future calls.
- Start each meeting by giving thanks for the feedback you got last time and highlighting a suggested change you're going to make for this call.

Provide rich content at a low cost of admission

Be clear about what you're offering participants and what you're asking of them. While people are working from home and balancing their time and responsibilities, it's important to help them make informed decisions about events they attend online and those they keep up with by other means (by newsletter, for example). You should:

- Be clear about the platforms you're using and the steps you've taken to make them as safe and accessible as possible for your participants. Consider co-facilitating with someone else to build some diversity of voice and representation into the sessions, and alternate facilitation and

platform-related issues or trouble-shooting duties during a call.

- Limit the duration of your calls and your role in them. Don't schedule meetings longer than 45-60 minutes unless everyone working from home expects a more in-depth, workshop-like experience. Also, limit the amount of time you spend speaking or delivering content in favor of holding time and space for community conversation.
- Design your calls and events to be low-prep or no-prep. Rely on peer-to-peer learning to provide most of the value in each call so that people don't have to study up or do homework to participate.

Stay engaged

If you're curious about more online facilitation and community management programming like Movement-Building from Home, you can keep up with all the latest news and offerings from the MozFest team on Slack[22] or via our newsletter[23] and on social media by following @mozillafestival. You'll also find recordings and notes from our calls on the Movement-Building from Home playlist.[24]

It's crucial that we improve how we connect with each other online and take what we learn with us into a more humane and participatory digital future. That better future is one that we can only discover together.

22 https://mozillafestival.org/slack

23 https://mozillafestival.org/newsletter

24 https://www.youtube.com/user/Mozilla/playlists

What I learned about working openly after one year on a distributed team

Anupama Jha

I work on a communications team that's part of an organization with an open culture, one where associates have the freedom to choose how, where, and when they work best. Working in this role has been my dream for a long time, and it's my pleasure to work with such a keen and passionate group of people. And it's the first time I'm working on a fully distributed team, one spread across continents.

I've been doing it for a year now, and I can see why distributed teams (and distributed teamwork) are becoming increasingly popular: they often lead to more productive contributors and organizations, especially when people are primarily working by themselves. I also understand firsthand the challenges these teams present: ensuring important details don't getoverlooked because of communication nuances, sustaining motivation, keeping up-to-date on work priorities and in sync with teammates, etc.

But the biggest challenges lie in communication and collaboration. Luckily, open organizations excel at addressing these issues.

So in this chapter, I'd like to share my experiences after one year on a distributed team—in the hope that I might help others who have found themselves working in a similar way. In particular, I'll discuss what *individuals* can do to make a difference in and have an impact on their teams and organizations, regardless of geography. Sometimes, I think we're so focused on making changes

on a broad, organizational level that we may forget how these changes happen *at an individual level first*.

Fortunately, open organizations have for some time experimented with methods for building rapport and fostering collaboration across distributed teams. I'll discuss several here.

Facing doubts

Starting out on a distributed team might feel strange. It might even feel isolating, as you won't always have an easy way to get to know everyone on your team quickly. I know I felt it initially. In fact, I'd even reached a point where I began questioning whether or not I was a good fit for the team.

I was in one of those deep pits of self-doubt (you know, the ones where you question everything you say and do). But to me, doubting yourself is a good thing. Every new challenge brings a new level of self-doubt. So self-doubt isn't a sign of weakness; it's a sign that you're being challenged. Once you notice it, you can even draw strength from it. It's natural.

Even so, my team exhibited a certain quality that helped me face my self-doubt directly, learn, and evolve as a team member and an individual. Always remember: Get noticed for the right reasons, and settling in will be easy. That seems especially true in the initial stages of joining a distributed team, when you are experimenting and setting your direction.

So many lessons during my first year of distributed work have shaped the way I think about becoming a better team contributor. One of the most important lessons is also one that helped dispel my self-doubt: Leadership is recognizing that *we are all one*, that every person you lead is as brilliant as you and has the same capacity for growth and accomplishment (despite their doubts). Rather than guarding your insecurities, share your doubts and seek suggestions. The sooner you can let go, the sooner you can start thinking about the new problems you must solve.

Communication management

Distributed teams won't work without the proper resources, technologies, and support. Thanks to the growing popularity of open source applications and instant messaging tools that streamline communication, collaborating online is easier. These technologies help bridge distances and create a workplace that supports associates working remotely. Every bit of work, information, and relevant conversation is recorded and stored together.

So one of the most important aspects of working on a distributed team, I've learned, is communication management. Communication management is like gardening, and with poor communication management, the weeds (the messages, the threads, the reminders, and more) can easily overtake a distributed team. Hence it's always important to establish good communication habits when working across different time zones and daily schedules.

Sometimes, more isn't better. Having too many communication methods becomes chaotic for a team, and every additional channel threatens to drain our focus. To keep all those communications from overrunning you like weeds, it's important for distributed teams to identify the team's "essential" modes of communication.

That is, it's important that the team work to establish *clear expectations* of use of *each* communication channel in use. For instance, we've designated Google Hangouts for synchronous chatting or urgent conversations. We've also designated WhatsApp for synchronous messages that aren't pressing (like informal conversations).

We certainly haven't solved *all* potential communication issues or addressed *all* potential communication scenarios, but we *have* developed a communication culture that values investing time in *writing and documenting as much as possible* (see the next section for more on this). We use project tracking tools (like Trello) to manage our projects and maintain ongoing communication on a regular basis.

In order to manage the volume of communication and to tease out nuances, we have individual, one-on-one meetings, and we also get together once each week for a touch-base meeting. Collectively covering all of a project's details is helpful for all of us. These meetings also enable us to set collective goals so that we're effective with our actions and not overwhelmed.

Email, chat, and video conferences, are essential for maintaining healthy relationships across the team—and for understanding whether work is going well or poorly. A lack of shared physical environment can restrict the information your team might otherwise share if it worked together in an office. Always keep in mind all the information *you* have—but your teammates *don't*—about how you feel and what you might need.

Being transparent about what you need can be difficult, but your peers need all the information you can give them if they're going to work with you in the best way they can. And not every interaction should be focused on "sharing information." Sometimes, you'll want to invest some spare time into informal talk about non-work topics. This can help your team develop its bond.

In short, having a good communication culture is key for a successful distributed team and an important factor in team productivity, whether your team is remote or not.

Visualize the work

One of the biggest hurdles to overcome while working in a distributed way is team members' lack of immediate visibility into what others are working on. To combat this challenge, you'll need to ensure you have a clear view of everything your team (and individual teammates) are doing. And you'll need to understand what *every team member does*.

Collaboration is critical here, especially when you're just getting started as a new team member. Collaboration requires your team's entire system: technology, processes, organization, and the people.

So how can you ensure proper collaboration on a distributed team?

As I noted earlier, communicate frequently (both within your team and with individual members), and regularly explain your progress on goals and objectives, thus avoiding ambiguity. Creating visibility through transparency is vital for building trust, and trust is central to effective collaboration. When we talk about trust on a distributed team, we're not talking about trusting that people are working "9 to 5 hours," are making daily check-ins, or anything like that. In this case, trust is more about *values*—trusting that everyone on the team shares the same values and sense of direction. That helps empower all members to manage their time and their responsibilities in the ways that are best for them.

My team trusted me from the outset, and that helped me build my working relationship with them. With their support, I was able to quickly adopt the team's objectives and its overall strategy. I was able to embrace open culture and implement its principles in my geography. My advice here: Learn how your work fits into your team's charter, understand what you can do to improve the overall picture, and make sure you've recorded and shared this with your team.

Avoid being burned out

Before working on a distributed team full time, I worked as an intern for an advertising agency in a "creative under pressure" environment. For me, the environment was stressful; I lived constantly being chased by deadlines. I was working a lot of hours, but I wasn't really enjoying what I was doing. Eventually, I burned out. But the source of that burnout was obvious. I'd squeeze in work whenever I had a free moment, and that led to some bad habits I'm still trying to break.

Being on a distributed team makes me focus on *quality over quantity*. I have to make sure I'm maximizing my seven or eight daily working hours, instead of just aimlessly working more than 12 or 14 every day. My team maintains a kanban board (visible to

all members, of course), where we record our top priorities. Every Friday, we collectively return to that board and analyze the week. What are the things we're happy about? What are the things that could have gone better? Have we done the most important work we could do the whole week? Collaboratively setting our intentions and returning to them weekly keeps us focused—without burning out. Perhaps this specific method won't work for you or your team. Regardless, find *some* way to measure your input and output for the week in a form that motivates you.

One more note here: When you're working on a distributed team, there's no one sitting next to you to tell you to go home, no signals that the office is closing—so *you* must be the one who decides when to stop. Having a "hard stop" is important. Flexibility and work-life balance are benefits of working on a distributed team. But you'll need to be the one to enforce that balance.

Looking forward

Everyone on our distributed team has our own quirks, our own responsibilities, our own skill sets, our own roles—but in the end, we're a team, one that embraces open source culture and influences other associates across our entire organization. My teammates are based in different countries. Every week, we're able to share a problem, share success, share a laugh, and share a story as a team.

And I still haven't met them in person. Does that matter? Does that affect our work? Does that affect our customer relationships?

Absolutely not.

Irrespective of location, it's our virtual *team* who thinks, acts, and works exactly how we are supposed to—without being in the same room. What's relevant is that every single person on the team knows why they're there, has a clear direction of their role and responsibilities, has the tools required to steer them in the right direction, and has the full backing and support of everyone else.

Leading and managing

Could your team be managing itself?

Alexis Monville

Now that many people are working from home and their teams are distributed across the world in what seems to be a sort of "new normal," managers have an important question to answer: Could your team be managing itself?

I can already imagine some of you, dear readers, frowning and thinking: "A team *has* to be managed."

As issues of distributed teamwork and management are more critical than ever in this new context, let me try to clarify why I believe it's essential to discuss designing organizational roles that won't become bottlenecks (roles that won't prevent the organization from delivering efficiently or to adapting quickly to changes).

In classic organization design, we tend to think that designing boxes on an organizational chart and putting great people in charge will solve all our problems. That approach *could* work in static environments, where what you have to deliver is defined once and for all. But if your environment is continually changing, you need to adapt your value proposal to those changes. Your organization needs to be adaptable.

Let's say that you're on the path to designing "the boxes" of a new organization. On your radar you could have managers that will assume full responsibility for certain groups and team leaders with full responsibility of the teams making up those groups. Static groups, static roles, static functions.

But you can't achieve a capacity for adaptability in your organization if you rely on overloaded people dealing with multiple

responsibilities. I suggest an alternative: Self-organizing teams designed around roles that aren't bottlenecks, roles that team members could take either full-time or for a portion of their time.

Please don't jump to the conclusion that my goal is to remove all managers and team leaders from the organization—as if self-organizing teams and management are somehow mutually exclusive.

Not exactly.

Managing the self-organizing team

The Open Organization Definition[25] lists five characteristics as the basic conditions for an open organizational culture:

- Transparency
- Inclusivity
- Adaptability
- Collaboration
- Community

I've recently discussed the importance of making work visible when attempting to achieve transparency and collaboration at scale.[26] Here, I'm more concerned with *adaptability*—creating teams without single points of failure, teams better able to adjust to changing conditions in dynamic environments.

I agree that a team has to be managed, and I think many of the activities we see as the sole responsibility of managers or the team leaders could, in fact, be delegated directly to the team—or to team members that could effectively deliver the activities *serving* the team.

So from my perspective, a team *has to be managed*, but a large part of that management could be done *by the team itself*, creating a self-managed team.

Let's review some of those activities:

25 See Appendix

26 https://opensource.com/article/18/7/high-impact-teams

- understanding the business and the ecosystem the organization evolves in
- understanding why we provide solutions, products, features, services and formulate a clear vision
- defining what needs to be delivered and when it should be delivered
- determining how it will be architected
- identifying how it will be implemented
- defining how it will be documented, demoed, tested
- distributing the work between the team members
- delivering the work
- implementing the documentation, testing
- presenting the demo
- collecting feedback from users and stakeholders
- ensuring that the result of the work is continuously flowing to the customers or users ensuring that testing is automated and triggered for each and every change
- improving the way the team is working and increasing its impact and sustainability
- improving the efficiency of the larger system formed by the different teams
- supporting customers and partners who use the product
- fostering collaboration between users, customers, and partners in the defining and implementing the product or service
- defining the compensation of team members
- controlling the performance of team members
- supporting and developing people in the team
- hiring people

When I look at those activities, I see some that could be delegated to a system put in place by the team itself—like the distribution of work, for example. The distribution of the work can be made obvious for team members by simply making the work visible to everyone.

I also see activities that are *difficult* to move away from managers, like managing the compensation of team members (because it would require building a compensation system that's more transparent, which is difficult when you don't start from scratch due to preexisting discrepancies in people's compensation).

I see activities that are focused on *users* and the *why and what* the team is delivering. On some teams with which I've worked, those activities are delegated to a team member taking, for example, the role of User Advocate or Product Owner (to use the Scrum terminology).

I see activities that are focused on *how* the team is working. Those activities are delegated to a team member taking, for example, the role of Team Catalyst or Scrum Master.

In both cases, their role will be to ensure that the activities are done by the team, not necessarily to handle everything by themselves.

By looking at the activities in more detail, I can envision many of them being handled by team members as part of their current role, or in a new role.

Giving managers or team leaders the ability to consider the activities for which they're accountable and the activities they can *delegate to the team* can remove the bottlenecks and single points of failure that currently exist in some organizations.

Having clarifying conversations about various roles on the team is even more critical when the team is distributed. It helps people performing the different roles understand the bigger picture—and free them to propose their help when something is going sideways.

One last thing. Getting started is easy. Open a shared document online with your distributed team. Ask team members to describe what they believe team members' roles and responsibilities are. Then invite team stakeholders to contribute too. You'll see discrepencies. You'll have disagreements. Resolving them *as a team* will make the team that much better.

About the contributors

Peter Baumgartner is the founder at Lincoln Loop, a web agency specializing in the open source Django web framework. He is constantly learning and is well-versed in many technical disciplines including DevOps, scaling, back-end, and front-end development.

Bryan Behrenshausen is

Ben Cotton is

Isabel Drost-Fromm is

Laura Hilliger is a writer, educator and technologist. She's a multimedia designer and developer, a technical liaison, a project manager, a conceptual architect, an open advocate who is happiest in collaborative environments. She's a director of We Are Open Co-op, an Ambassador for Opensource.com, is helping to open up Greenpeace, and a Mozilla alum. Find her on Twitter and Mastodon as @epilepticrabbit.

Anupama Jha is an internal communications strategist at Red Hat and is based out of Pune, India. She pursued communications and public relations as her academic specializations, and is involved in volunteering at various NPOs, which helped her to develop leadership skills, become a world citizen, and empower others.

Guy Martin is the Executive Director of OASIS Open, an internationally recognized standards development and open source projects consortium. He is responsible for the organization's overall operation, in addition to helping define its cohesive strategies and policies to deliver the best value to its members.

Alexis Monville is building high impact sustainable organizations. Alexis is a member of the Engineering Leadership Team at

Red Hat. Alexis brings more than 20 years of operations and management experience. Over the years, Alexis worked in diverse sectors, from the automotive industry, the epic Web start, IT consulting, public sector, software development, which led him to found a management and organization consulting and coaching firm.

CHAD SANSING works on leadership and training, as well as facilitator and newcomer support, for MozFest. When he's not at work, you can find him gaming, reading, or enjoying time with his family. Prior to joining Mozilla, he taught middle school for 14 years.

CHAD WHITACRE was the founder of Gratipay, a funding platform for open source software developers and other creators. Gratipay's open company vision lives on through a successful fork, Liberapay.

Appendix

The Open Organization Definition

The Open Organization Ambassadors

Learn more

Additional resources

Get involved