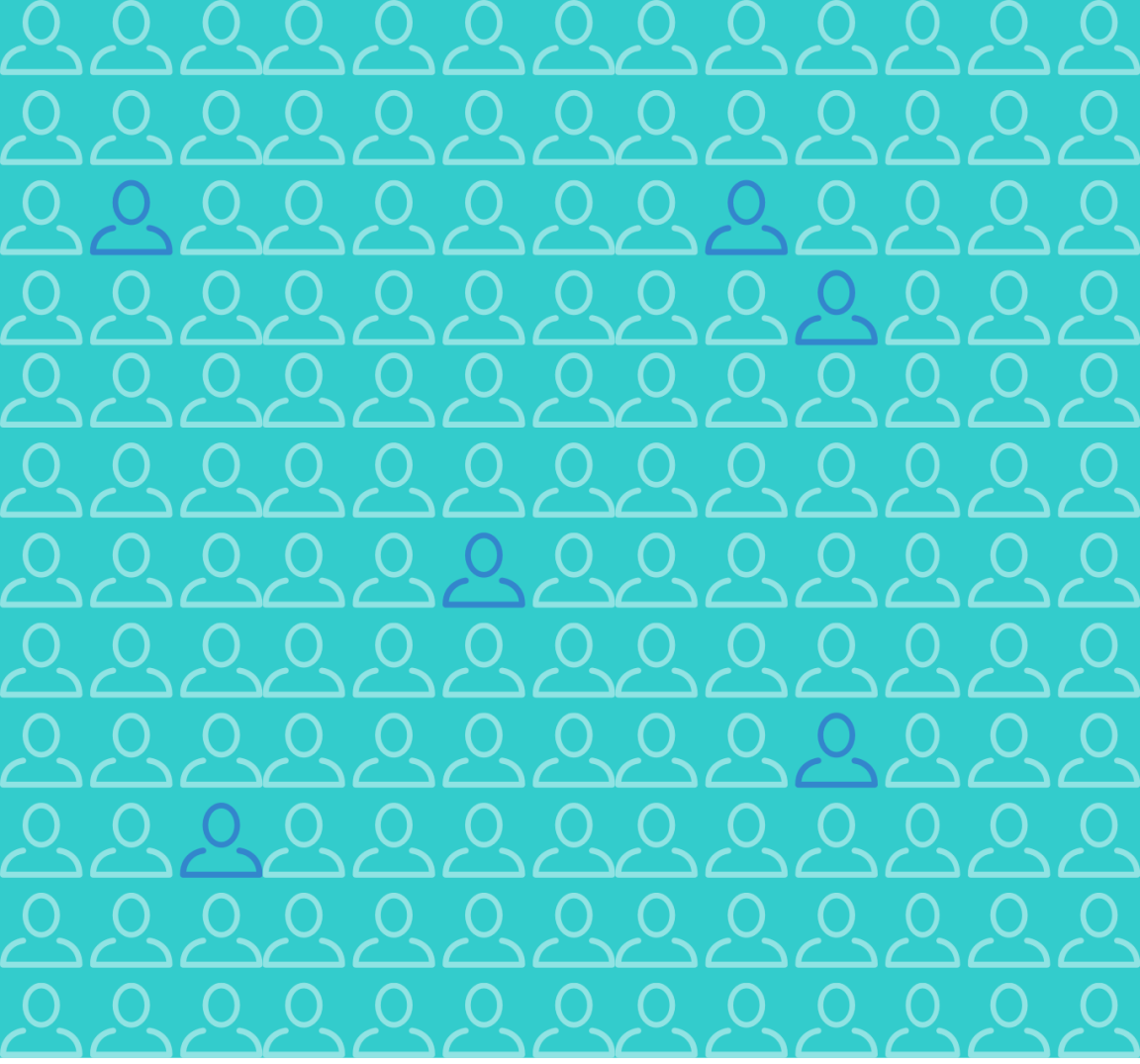


Human at a Distance

An Open Organization Guide To Distributed Teamwork

Open Org

Human at a Distance

Human at a Distance

An Open Organization Guide to Distributed Teamwork

Copyright

The following chapters were originally published (in whole or in part) at Opensource.com:

- Peter Baumgartner's "The importance of trust on distributed teams"[2]
- Laura Hilliger's "How to run an online community meeting in 11 steps"[3]
- Guy Martin's "It's not the tool: Building a culture of transparency through company-wide chat"[4]
- Alexis Monville's "Understanding self-organization and management"[5]
- Chad Whitacre's "Distributed teams benefit when they track issues publicly"[6]

---

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

2 https://opensource.com/open-organization/17/7/lincoln-loop-trust

3 https://opensource.com/open-organization/16/1/community-calls-will-increase-participation-your-open-organization

4 https://opensource.com/open-organization/17/12/chat-platform-default-to-open

5 https://opensource.com/open-organization/18/8/self-organizing-team-management

6 https://opensource.com/open-organization/17/2/tracking-issues-publicly

Colophon

Typeset in DejaVu[7] and Red Hat.[8] Produced with LibreOffice.[9] Cover design by Laura Hilliger.

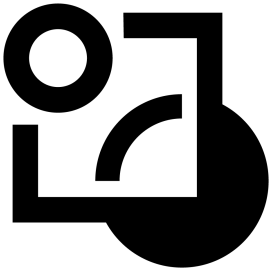
7 https://dejavu-fonts.github.io/

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

9 https://www.libreoffice.org/

Version 1.0

December 2020



Also in the series

The Open Organization Field Guide: Practical Tips for Igniting Passion and Performance

The Open Organization Leaders Manual: A Handbook for Building Innovative and Engaging Teams

The Open Organization Workbook: How to Build a Culture of Innovation in Your Organization

The Open Organization Guide to IT Culture Change: Open Principles and Practices for a More Innovative IT Department

The Open Organization Guide for Educators: Transformative Teaching and Learning for a Rapidly Changing World

Contents

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.[10] 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-

10 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.[11] 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

11 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,[12] so you can see the work as it occurs and provide us with your feedback—or, even better, your own contribution.

12 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 seven time zones.[13] We have no central headquarters. All members of our core team can see all our financials (literally every penny earned or spent)[14] and choose their own salaries.[15] 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

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

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

15 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.[16]

16 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 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.

Building cohesive remote teams

Sim Zacks

"*Science may never come up with a better office communication system than the coffee break*."—*Earl Wilson*

A team is a group of individuals working on interrelated tasks. And most teams seem to encounter similar problems:

- At points of integration between different tasks, miscommunication about responsibilities may occur
- A team member with more experience on a certain aspect of a project might not get assigned work that aligns with that experience
- Team members might miss opportunities to help one another with their work, causing issues with efficiency

These issues don't disappear when teams are distributed geographically. In fact, they can be exacerbated. Success as a distributed team—like success with *any* team—is more certain when that team works together as a *cohesive* team—a team whose capabilities exceed the capabilities of all individual members. A cohesive team consisting of average-skilled employees can easily outperform a non-cohesive group of superstars. To get to this level, where "the whole is greater than the sum of its parts," each individual contributor must work in an interlocked fashion with other contributors.

In this chapter, I'll explain the following seven qualities that lead to cohesive teams:

- Understanding the team goals
- Informal interactions

- Knowing the whole person
- Communication
- Diversity
- Regular feedback
- Effective retrospectives
- Celebrating success

Perhaps unsurprisingly, each one is in some way related to interpersonal relationships. And when team members think and act openly, each one of them improves.

To build a cohesive team, you have to work on all the aspects that surround work, though they are not necessarily part of the team's workload. Each one of these dimensions raises its own challenges when the team is working in a remote environment—and even more so when the team is distributed globally.

Understanding the team goals

Everyone on a team has individual tasks to complete. However, the more each member understands the bigger picture - what everyone else is doing, and why - the easier it will be to shift direction if they need to. It will also be easier to ensure that the solutions they're developing are indeed the best solutions (and the more opportunities they'll find to add unanticipated value to each other's work).

So how do team members discover others' goals?

A team physically co-located can derive a great deal of information through osmosis—overhearing spoken conversations, seeing what people are drawing on the whiteboard, and talking to people over a coffee or smoking break. They can ask and answer questions, exchange valuable context, and their understanding of what the team is working on slowly emerges.

Because remote team members don't always *see* other team members doing their work, they don't receive that osmotic information, and they aren't guided as much by natural curiosity. So a remote team must consciously plan to disseminate the information that'll pique curiosity.

However, since this kind of osmotic information isn't always directly related to the work that each person on a remote team is dealing with, certain members might feel it's irrelevant—and filter it out. So teams need to deliberately reinforce osmotic information flows and constantly ensure that it's tying each person's specific work to the team's goals and objectives. During meetings, for example, it's important to ask everyone providing a status to also relate to their work in terms of the team's broader objectives. This will help *everyone* focus on the team's joint mission, and not get bogged down on the details of their specific tasks.

Informal interactions

In distributed organizations, people are often assigned tasks appropriate to their professional profiles—not necessarily based on their personal abilities. So teams might not be able to infer the full extent of their teammates' talents and interests simply by observing the work they're doing. Therefore, creating ample opportunities for information interactions between distributed team members is essential to team cohesion.

First, doing this can help team members locate and draw out hidden talents on the team. Teams are comprised of individuals, each with their own strengths and weaknesses, even when they all qualify for the same job description. Remote teams—all teams, really—are successful when they draw out their members' strengths while minimizing weaknesses. In this way, every team can tap individuals for the aspects of the work to which they can best contribute. One person may be better at external communication, another at learning new things quickly, and a third at graphic arts. These attributes might not be readily observable. Rather, team members learn about them from one another through interpersonal interactions, perhaps while taking coffee/smoking breaks, eating lunch together, or going to the bar after work.

Fostering informal interactions on a distributed team is also important for helping that team collaborate more effectively. In short, they're more likely to work better together when they under-

stand and recognize one another more completely. Here's one example of what I mean: As a member of a team, you often have to trust your teammates and understand what they mean when they say something. Someone who says "I'll have that done by tomorrow" might mean one of several different things by that sentence from a practical perspective. Some people will have it done on time (or before), and if you ask them about it beforehand, they will consider that annoying. Others need reminders and a bit of pushing if they're going to deliver the work to you as promised. On a distributed team, you'll only understand nuances like these if you've spent some time really getting to you know your teammates and understanding their personalities. The more you know about your teammates, the more you will be prepared for anything they throw your way.

Having a personal relationship with people in a remote/global environment can be more challenging, as you must explicitly build relationships instead of having them slowly develop organically by virtue of simply occupying the same physical location. One strategy for fostering these relationships is schedukling regular coffee breaks. In this case, a "coffee break" is a block of time in which a few people get together on a video call for specifically non-business conversation. It might seem a bit contrived at first, but with time, you'll find it helps build and cement interpersonal relationships.

Communication

In face-to-face environments, people have access to multiple means of communicating. Tone, facial expressions, and body language are just as important (if not more so) than the words people use when interacting. These channels are rich sources of meaning, and learning *how* your teammates convey ideas is critical to understanding what those teammates are saying. ("The most important thing in communication is to hear what isn't being said," Peter Drucker once noted.)

Yet on distributed teams, faceless communication—facilitated perhaps with email or text chat—so much of that nuance can be lost. And opportunities for misunderstanding can multiply.

When communicating with a remote team, try to keep in mind the following ideas:

- Each communication medium has plusses and minuses; try to find the best medium for your specific communication goals
- Assume good intent; it's possible you aren't reading the message in the same "tone" it was sent
- Try not to use sarcasm, cynicism, or other communication styles that require the kind of nuance that can't be easily understood through some channels
- Use emojis to convey tone, but keep it simple and professional
- Use video and interactive dialog to keep people engaged, especially in remote environments where distractions abound

Diversity

Every person is unique and has their own way of viewing and interpreting any situation. Everyone's perspective is generally colored by their life experiences, education, and neurodiverse cognitive patterns. A team composed of people with similar backgrounds will have a much narrower range of perspectives than a team made up of people with different backgrounds. This diversity enables the team to look at a situation through a multi-faceted lens, giving them a wider degree of understanding of the big picture.

Diversity is one of the few components of cohesive teams that is potentially *stronger* in a remote environment. That is, distributed teams can have members anywhere; teams comprised of people from multiple backgrounds, countries, and cultures have the ability to work together and increase the value they provide. A team whose members learn to interact with each other and feed off

of each other's strengths will be able to blend into a cohesive mesh.

Regular feedback

Feedback is one critical aspect of a team's ability to constantly improve. Team members must understand what they're doing right so they can *continue* doing that. They must also understand the mistakes they've made so that they can work on those issues. Feedback can come from multiple source and travel in multiple directions. Managers who oversee a problem with people working together can suggest methods of improvement. Teammates can make suggestions to each other regarding problems they've seen and suggestions on how to improve. Customers can provide feedback on the improvements that will improve their lives.

Managing these feedback loops is much more difficult for remote, distributed teams. Fewer interactions between team members means certain people might not feel comfortable giving constructive criticism. (See the section above, on informal interactions!) Leaders and managers don't always observe team member interactions and may not be aware that some of the team members are not working well together. The feedback loop has to become an object of deliberate structure and care (more than, say an informal discussion in the cafeteria).

To accomplish this, team leaders and managers should hold regular one-on-one meetings with team members. To be most effective, this should include a review and status update of both task- and career-based objectives. During these meetings, leaders should provide feedback about the tasks for which the team member is responsible and ask team members for feedback on those tasks, on the team leader, and the team's general direction. This should be an open conversation, where all parties feel safe and valued.

Effective retrospectives

When a project reaches a milestone—or wraps up—it's generally a good idea to conduct a retrospective (or "post-mortem") to review how the team succeeded on the project and how it may need to improve in the future. This involves analyzing the *entire* project process—both the "good" aspects as well as "bad." An effective retrospective process enables team members to provide feedback to their peers and for everyone to learn from one another. During this retrospective, each team member should learn about methodologies other team members used. Comparing methods this way will often highlight strengths and weaknesses, as well as gaps and collaboration opportunities. By learning all of this, and building bridges to mitigate problems for the future, you'll greatly increase the team's cohesiveness.

In a remote environment, a retrospective should be a multi-step process that looks something like this:

- Survey
- Review and follow-up
- Meet and discuss
- Action items

Survey

Ask participants to collaborate on writing questions to ask the team about the project. Each retrospective will be unique and require its own questions. But here are a few examples you can customize: - Was the initial project definition sufficient for understanding the scope of the work? - During the course of the project, what did the team learn that it should have known in advance? - Were there times when the project could have used more collaborative efforts? - What would you do differently if you were starting the project from scratch today? - Was the decision-making process transparent, and did you have the ability to influence the way the project unfolded?

Your goal is to ensure that everyone's ideas and concerns are taken into account and included. After the team agrees on the

questions, they should compile them and send them to all team members in the form of a survey. As much as possible, every question should lead to *actionable* outcomes; based on the group's answers, that is, the team's process should *change* during its next project, if necessary. Moreover, when possible, questions should have fixed options as answers. This will enable you to generate statistics per answer. Your survey should also include a place for additional comments, so everyone has the opportunity to provide their complete feedback.

Review

When all survey results are complete, publish them for the entire team to see and review. Leaders should address any questions about the results so everyone on the team understands them. Everyone on the team should keep track of their own comments on each result.

Meet & Discuss

After the individual review is complete, it is time for the retrospective meeting. During the meeting, review each question and allow everyone on the team to share impressions and comments. The result of this session should be a list of the aspects of team workflow and process the team feels should change in future projects (and those that should remain the same).

Action Items

Send the finalized list to the team for an additional review to gain final feedback before declaring the process complete.

Celebrating success

Celebrate when the team successfully completes a project or process. Doing this not only reinforces a culture of positivity but also enables the team to spend time together, informally socializing.

For co-located teams, this might involve going out for drinks or dinner after work, sharing a cake, or any other type of celebra-

tion that brings the team closer. Moments like these enable the team to relive the victory and discuss future plans.

On a remote team, celebratory traditions like this are more challenging, as it can seem like members are celebrating on their own. To espouse a culture of positivity on a distributed team, consider beginning regular team meetings by asking members to share something they appreciate. This provides every participant the opportunity recognize someone else on the team who did something positive. This peer-recognition is a great way to reinforce every team member's value. If the team has a budget for awards, it might consider initiating something like a "member of the month" award (or similar), for which the winner is sent a small token of appreciation (like a restaurant gift certificate). And if every member of the team is working in the same (or closely aligned) time zones, the team might gather on a video call at the end of the work day and share a drink.

Expect challenges

In conclusion, there's no doubt that developing a cohesive distributed team is more challenging than it would be in a co-located environment. However, by thinking through all the stages and keeping this goal in mind while establishing workplace conventions, you can acheive similar results.

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.

Communication practices typical of high-performing remote teams

Jimmy Sjölund

One often-cited advantage of a colocated team—everyone sitting together in the same space—is the ease of instant interaction and collaboration. Need something? Just turn around and talk to your teammates! In fact, the Agile Manifesto even seems to enshrine this kind of interaction:

> The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. But authors of the Agile Manifesto were working in 2001, when remote collaboration was much more rare than it is today—and when many fewer tools for distributed teamwork were available. While face-to-face conversation is still very effective for achieving communication with team members, so much more is possible now through "eFace-to-eFace" interaction.[17]

So how can you translate the colocated experience to a remote setting? Don't expect to easily replace the act of turning to your colleague with pinging them in a chat tool. You'll need to consider much more when working remotely than simply translating "old" behavior to a "new" setting. You might even need to find communication techniques *better* suited to a remote context.

---

17 https://agilemanifesto.org/

Over the years, I've worked with several teams this way—and I've noted the practices that have made them successful. In this chapter, I'll share some of those practices.

Synchronous communication

First let's examine a few useful techniques for *synchronous* communication. Synchronous communication is communication that occurs between two or more persons in real time. In synchronous settings, there are no delays between the persons taking conversational "turns," and all parties can talk at the same time (though that's perhaps not the best communication style!). Holding a conversation in the same room or an audio/video call are examples of synchronous communication techniques. Typed chat conversations can also be synchronous, as in cases where all parties are "live," reading and responding to typed messages as they appear. (If the chat tool has a "history" feature, then others can read what has been written before and respond to that later on—but that would be asynchronous communication. More on that later.)

The watercooler

Much of communication and networking in the workplace come from meeting people at the office. Many discussions, plans, and collaborations have started (or even been decided and concluded) in informal talks over coffee, at lunch, or in the corridors. Sometimes, people refer to this as "meeting at the watercooler" (perhaps a thing of the past—but the term has stuck around). These mini-meetings are often spontaneous—but when you're working in a remote, distributed fashion, you need to plan even the smallest synchronous interactions (therefore removing the spontaneity!).

Successful teams and organziations recognize the advantages of these interactions, and one solution they've developed is setting up a "watercooler channel" in the chat tool of your choice. This channel is the place where you simulate the colocated experience of casual, unexpected conversations. There, you can take part

some of the idle chitchat (or business talk!), or just hang out while taking a small break. Some organizations establish their water-cooler channel as a team-wide chat, others as a department- or office-wide channel. Why not combine them all and host everyone? Most importantly, make sure the team knows that business-critical chatter should not happen here.

You'll also want to create an "information channel" for managers who need to share information and deal with the "this meeting could have been an email" phenomenon (see below). This room's just for socializing.

At the company where I work, for example, we have a network you can join to be paired up with a random colleague for lunch. This has now migrated to the distributed setting, where instead of going to lunch together, you and a random colleague have lunch and a video meeting occurring at the same time.

The day-long meeting

Not another meeting?! Don't worry. It's not. Not *really*, anyway.

Some teams have had great experience with hosting an ongoing video meeting—an "open room" that's always activated for anyone to visit when they want to chat. This practice involves creating an "all-day" meeting in a video conference platform of your choice, then distributing the dial-in details to everyone on the team who might want to access it at any time. Unlike the watercooler channel, this one is typically more business-focused. When you run into to an issue or want to discuss something with a co-worker, you can tune into the all-day meeting and see who is available. It's an alternative to the practice of turning around and asking your colleague in the room a question; the video call makes the communication more personal and efficient than a text or voice-only chat.

Asynchronous communication

The opposite of synchronous communication is *asynchronous* communication, or communication that occurs with the

conversational "turns" delayed to some degree. Typical examples of asynchronous communication practice include sending letters or, more recently, logged a message for someone in your team's chat channel so they can read it in the chat history later (as I mentioned above).

Standups

Teams that work with lean and agile practices usually run some kind of "standup meeting" on a regular basis (some do it every morning, and others less frequently). At this meeting, everyone gathers around the team board and looks for work-blocking impediments they might need help with, or to see if the team needs to adjust its priorites priorities somewhere. A common formula for a report at the meeting usual looks like this: quickly talk about what I worked on yesterday, discuss what I will focus on today, and explain any problems the team needs to address before I can do that work. A team standup usually runs for 15 minutes, give or take. Performing a standup in a remote-first setting it can be more difficult, especially when you can't find a convenient time to gather everyone in the team in a synchronous channel (even if you are all in the same time zone, team members will likely have barriers that prevent them from meeting at the same moment!). To me, one of the primary advantages of remote work is the possibility to balance work and life a bit better. Therefore, consider the strategy of *not* running the standup as a synchronous meeting where everyone must attend in person at the same time. Instead, take another opportunity to use a channel in your team's chosen chat tool. Everyone can prepare their typical standup statements in advance and write them in the channel. If you're reporting on impediments you would like to discuss, or something on which you need support, you can simply arrange those meetings afterward with the appropriate group of people.

Colocated standups—everyone interacting at the same time—do have benefits, and some argue that asynchronous standups are mere "status reports" and not an effective communication activity. Why not try it and see if it works for your team?

Sharing information

Many meetings are held for the purpose of sharing information. One-way information distribution remains common on distributed teams, even while not all information needs to reach everyone at the same time (or within a specific time). But maybe your team doesn't need another video meeting to do that? I encourage leaders to create a separate channel for information to the team.

One common mistake teams make when shifting from a colocated environment to a distributed one is to introduce more meetings to compensate for the social dimension of being in an office and the spontanious chats and discussions that occur there. But this can prove to be a blessing in disguise, as your team might end up with online meeting fatigue.

Instead, try to work with an information channel in your collaboration tool. There, you can post post the information that you otherwise would have raised and recounted at, say, a weekly meeting. Doing this has another advantage, too: by posting the information in a dedicated instead of sending an email, you're gradually creating a shared knowledge repository others can easily search and help one another find. Everyone can interact with questions or comments and, as most collaboration tools support threads, you get all the questions, comments, answers, and discussions available and searchable in one place. That way, if several team members have the same questions, they can read the answers already given (rather than simply missing out if they didn't participate in the meeting or missed an email). This practice is also helpful for people who are more comfortable asking questions after taking some time to reflect on a topic, or don't always prefer to speak up in synchronous meetings (colocated or video enabled!). If you have a team spread over multiple time zones, this kind of asynchronous communication is even *more* important for information sharing.

A3 reports

A3 reports are the result of a work method or a problem solving approach, one that emerged from Toyota and the Toyota Production System, which one could say ignited the worldwide "lean" movement. Toyota uses A3 reports for many purposes—from status reports, to proposal and policy changes, to (most notably) problem solving. The name for this method derives from the paper size it once required: the largest possible size you could still fit into a fax machine (today, as less is more, Toyota also promotes A4 reports!).

While working remotely, whether temporarily or permanently, this technique for sharing information is even more effective, especially if you expect to might miss out on corridor chats and pick up information that might not have been communivated in a concise way. A3 reports that are openly available to all is a great way to share information and collect feedback.

The A3 report should contain all necessary information for handling a specific problem or making a particular decision. The format differs depending on whether it's an A3 for problem solving, a proposal, a status report, or for strategic planning, but in general it includes:

- A short description of the issue or background
- A status report of the current situation
- A description of future state or goals/targets
- An analysis of or a statement on the root cause (in case of problem solving)
- Suggestions for alternative improvements or solutions
- A recommended action, next steps or a time plan
- An analysis of the cost and value (if applicable)
- Follow-up plans

The A3 process is based on the Deming Cycle: *Plan, Do, Study, Act*. One of the advantages with an A3 report is that it makes plans visible and easier to read quickly, but one must remember it's also only *one step* in the decision-making process. Another piece is the *nemawashi* process, where the principle is to

do decision-making slowly and in consensus. Many people will circulate and discuss an A3 report before presenting it to decision makers or management. Feedback it receives along the way helps perfect the report so that when it's time to make a decision the information is clear and direct. This could potentially lengthen the time required for making a decision—but when everyone is on board, implementations are much faster.

It's imperative to stress again: *the A3 is not the tool*. The A3 is the *result* of the iterative process of analyzing the situation; it's the visualization method. In Toyota Kata, it's used in the dialogue between the mentor and the adept. The important thing is *how* and *why* an A3 is developed and used, not the format or the report itself.

A strategy-focused A3 is by nature future-oriented. It gathers the future goals and needs, and will cascade down through the organization to people's daily work. The visualization is important, and a well-crafted one-page message should communicate it easily. In the *nemawashi* process, many have already had the opportunity to give feedback, and through the visualization of the A3, everyone can quickly be reminded of the strategy and how their work is aligned with the goals and achievements.

Best practices (or 'One more thing ...')

eFace to eFace

If you have the options of adding video to your call or meeting, I encourage you to do so. With video, we can account for more of the feedback channels that colocated communication affords us —nuances lost when we're not in the same room, such as facial expressions and body language. Video can't *replace* in-person interaction, but it will be a great enhancement to audio-only communication. Similarly, don't use the option to turn off the camera as an opportunity to do parallell work; respect the time of the people around you. Be an active participant in the meeting.

That said, we're all different—introverts and extroverts, good days and bad days. In a remote setting, you'll quite commonly see

family members stepping into the room, or the (often much-appreciated) cat peeking in from the back of someone's laptop. So extend some compassion and understanding, too. Don't force your team members into turning their camera on at all times, but do appreciate when they do and try to do it as much as possible.

Don't over-compensate

As distributed teamwork proliferates, people are more conscious than ever about teammates who are working remotely and often work to compensate for the social interactions those teammates might be missing. But some tend to run the risk of *overcompensating* too. There are now post-work video happy hours, video lunches, video check-ins—we try so hard to make up for the remote situation that all the meetings and gatherings become exhausting. I worked, for example, on a team that, when shifting to a distributed environment, suddenly had twice as much check-ins than it had when we were at the office. Why? The meetings weren't providing value that they weren't before we made the switch; on the contrary they took more time away from other work. And if your team has never gone out to a post-work event while colocated in the office, don't arrange for several online after-work meeting now—unless, of course, the team specifically asks for it or arranges it themselves.

Agree on communications

When working with a team, one fo the first things I typically do is run a workshop that helps the entire team develop a "working agreement." Here, we propose things we want from a well functioning team and clarify what we like and dislike when working together as a team. Issues we address here include how to communicate, how we run meetings, how we conduct video chat, and what we expect of ourselves and our collegues. In a remote setting, this practice is even more important, as small things can grow into conflicts or discontent inside the team without you noticing as you might have done in a colocated setting. When we all agree on some ground rules and preferable behaviors, it's easier to get ourselves

back on track when we deviate: we just remind ourselves why we included something in the working agreement in the first place. There are several ways to run the workshop or reach an agreement. I've found inspiration in Jimmy Janlén at Crisp's Bootstrapping a working agreement[18] and *The Basecamp Guide to Internal Communication*.[19] Use these and other resources as inspiration and guide to your own way of working. Keep in mind: this should be a living document, something you return to and update; add new parts, remove obsolete parts, and make especially sure you update the agreement after the team the get new members.

These pointers and suggestions have worked for some teams I've assisted. But there's no guarantee everything I've discussed in this chapter will work for the *next* team I join. As teams are made of people, and as everyone is different, we need to adjust our ways of working and communication to every setting and situation. If something doesn't work with your team, move on. Try something else. Involve everyone. It's hard work. But when you succeed, you receive all the joys and benefits of a well functioning, happy team producing value and growing together.

Now go out there and make magic!

18 https://blog.crisp.se/2018/12/05/jimmyjanlen/bootstrapping-a-working-agreement-for-the-agile-team

19 https://basecamp.com/guides/how-we-communicate

What to select when you're connecting

Ben Cotton

Communication is key to any project's success. Recent discussions in open communities where I participate have made that clear to me. By and large, most of these discussions are about the merits of one communication tool over another. Should Discourse replace mailing lists? Should Telegram replace IRC? This chapter is not going to answer those questions.

That's because picking the right communication tool won't automatically make your distributed work successful. In fact, *there is no single "right" tool for your open team or organization*—just a set of considerations to think about in selecting communications tooling. The merit of any tool isn't intrinsic to the tool itself. It's a product of how well it solves a problem for your team.

Without a doubt, the tools you use to communicate will influence the effectiveness of your team's communication. But every open organization needs to arrive at a consensus about which tools works best for its workflow and culture, bearing in mind that the decision may attract some participants while driving others away. While some tools naturally favor certain kinds of interaction, your organization's culture matters more.

So let's review some considerations you might make as you select the tools for connecting your distributed team.

Categories of communication

In this chapter, I lump tools into two broad categories based on the kind of communication they facilitate: *synchronous* and *asynchronous*. Synchronous tools are designed for real-time inter-

action when participants are active at the same time. This includes various chat platforms, as well as voice and video call platforms. Asynchronous tools do not rely on participants being active at the same time. Email is perhaps the quintessential asynchronous platform, but forum tools, Kanban boards, and social media platforms are also good examples.

Of course, the categories blur a bit; you can reply to an email immediately and you can let a chat message sit until tomorrow. But most pretty obviously fall into one category or the other. Because you can bend a communication tool to cover both synchronous and asynchronous categories, you may be tempted to fit all of your communication into one tool. Picking "one tool to rule them all" is a valid option, but be aware that it immediately favors one category of communication over the other. Plus, the more a tool does, the less likely it is to do those things *well*.

Pick one

Let me introduce you to something I call "Cotton's Law of Communication": the number of communication platforms you use only increases.

Years ago, I worked for a company that used consumer Skype for both chat and voice-over-IP. It wasn't great for the purpose, but it worked well enough. Then someone discovered Slack, and we moved to that. Well, the engineering teams did. Marketing, sales, and administrative teams stayed on Skype. Then we started using Screenhero (later purchased by Slack) for screen sharing. Then Microsoft bought our company and we switched to Skype For Business and Microsoft Teams. But we also kept using Slack for a while.

Similarly, the Fedora Project has used IRC since the beginning. More recently, some contributors have started using Telegram, often bridged to the IRC server. And other contributors use Element, which *also* has an IRC bridge for many channels.

Twenty years ago, I chatted with my friends on AOL Instant Messenger. Now I chat with them on Signal, Google Chat, Tele-

gram, Whatsapp, Facebook Messenger, Twitter, Snapchat, Slack, and probably more that I can't even remember.

You get the picture.

Don't let this happen to your team. Pick no more than one tool for a given purpose. Otherwise, confusion reigns. And make those boundaries clear. Maybe it's an organization-wide decision. Maybe different teams *within* an organization are using different tools (fine, so long as everyone who wants to participate knows where to look). Just note that in the latter scenario any communication across teams will require every to manage multiple platforms—and there you go, increasing the number of tools you use.

Cotton's Law.

Do your best to keep entropy in check by intentionally choosing one platform and sticking with it.

What to select

In what remains of this chapter, I review some considerations you might make when your team is searching for a new communication tool. Generally, I'll present them as a series of tradeoffs with "X or Y?" framing and in no particular order. Your priorities are your own and you will have to weight each consideration appropriately.

The first section covers considerations that apply to both synchronous and asynchronous tools while the sections following that look at each of those categories in turn.

Considerations for all tools

Safety. Contrary to what I said above, I present this consideration first because you should indeed consider it first. You may want to think that everyone in your organization is great and will always behave themselves. The entirety of human history begs to differ.

At some point, you will need to deal with organizational safety issues via your communication tool. This is particularly true

if you use publicly-available tools (see below). The worst time to learn about your platform's safety features is after they're needed.

To ensure your tools promote organizational safety, I suggest reviewing Ben Balter's list of seven safety features[20] in which a reliably safe tool should have:

- blocking/muting
- reporting
- hiding content
- preventing new content
- community guidelines
- auditability
- user consent

While Balter approaches these issues from the perspective of tool *developers*, his list is also instructive for tool *selectors*, too. If the tool you're looking at does not include these seven features, you may want to reconsider whether it will meet your needs.

Self-hosted or externally-hosted. Do you have the resources—both human and financial—to maintain the tool yourself? If you do, then that's one way to maintain control over your tooling. But remember that time spent administering and troubleshooting communication tools is also time your team or organization can't spend working on whatever it is you do. Externally-hosted tooling (either free or paid) might provide less flexibility, but it's also isolated from internal infrastructure outages.

Open source or proprietary. This is entirely a value judgement for your organization. For some communities, anything that's not open source is a non-starter while others might not care at all. Most will fall somewhere on the spectrum between. Will your team or organization wish to modify, improve, or extend the tool it's using? If so, it'll need access to the tool's source code or design details, and an open source option might suit it.

Federated or centralized. Email is a federated system; that is, people using it in different domains (@gmail.com, @hot-

20 https://ben.balter.com/2020/08/31/trust-and-safety-features-to-build-into-your-product-before-someone-gets-hurt/

mail.com, @fastmail.com, etc.) can still communicate with one another. Conversely, centralized platforms *don't* allow that kind of cross-system communication. (Consider many contemporary social media platforms. Communication between Twitter and Facebook, for example, isn't possible.) Will your team, organization, or community need to connect with others using their own instances or implementations of the same tool? If so, a federated system might be desirable. But as communication tools today seem to be trending toward centralized models, you may need to work harder to find a federated system that meets your needs.

Public or private. Can outsiders see what you're saying? Should they? For many open organizations, public visibility is important. But even in those organizations, some conversations may need to take place privately (or semi-privately). Your open and distributed team will need to decide if (and how) to publicize certain aspects of its communication.

Archived or ephemeral. Do you want to be able to go back and see someone said last month, last year, or last decade? Do you want to maintain an easily referenced record of those conversations? Some conversations aren't worth keeping, but records of important decisions probably are. Your tool should allow you to meet your archival needs.

Third-party clients. Does the platform allow for third-party clients, or do you have to use an officially supported client from the tool's developer? People may want to use different clients to get a particular function the official tool doesn't offer. This can be an important consideration for people with certain accessibility needs.

Considerations for synchronous tools

Sometimes you just really need to talk to people in real-time.

Mobile experience. People do a lot on their phones, especially if they travel frequently or if their Internet service provider experiences an outage. What is the mobile experience like for the tools you're evaluating? (It's not just a matter of whether a tool ex-

ists, but whether it offers a desirable experience.) If the user disconnects while on an airplane, do they lose all the messages sent in their absence?

Status and alerting. What happens if someone stays logged in and walks away for a little bit? Do they have the ability to suppress notifications? Is there any way to let others know "I'm away or busy, don't expect an immediate reply"?

Audio, video, and screen sharing. Sometimes you need the high-bandwidth modes of communication in order to get your point across (or just shortcut a lot of back-and-forth). Does the tool you're looking at provide this? Is it usable by those who can't participate due to bandwidth or other constraints?

Integrations. Can you display animated GIFs? The ability to speak entirely in images can be either a feature or a bug, depending on the organization's culture. But if it's important one way or another, you'll want to make sure your tool matches your needs. Of course, there are other integrations that might matter too. Can your software build system automatically post alerts to your communication tool, for example? Does the tool automatically recognize certain links and display them in a particular manner?

Considerations for asynchronous tools

Of course, not everyone on your distributed team will be sitting at their desks at the same time. People go on vacation. They live in different time zones. They step away for 10 minutes to get a cup of coffee. Whatever the reason, you'll occasionally need to communicate asynchronously.

Push or pull. Email is a push mechanism. Your message arrives in my inbox whether I've asked it to or not. Web fora are a pull mechanism. I have to go check them (yes, some forum tools provide an email interface). Which works better for your workflow and community? Pull mechanisms are easier to ignore when you want to step away for a little while, but they also mean you might forget to check when you do want to pay attention.

Is it a ticket system? I haven't really talked about ticket systems/issue trackers because I don't consider them a general communication tool. But for some projects, nearly all discussion essential to the project and team success happens in a ticket tracker (like GitHub issues). If that works for you, there's no point in adding a new tool to the mix.

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,[21] and that any genuine and true openness is only ever built on a solid foundation of safety and consent. 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

21 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[22] to our legal structure[23] to our name,[24] 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.

---

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

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

24 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.[25]

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."

25 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.[26] 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.

26 https://github.com/gratipay-bot

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 com-

mitters 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.[27] I've co-facilitated barn-raising calls[28] for the Badge Wiki community[29] and educators

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

28 https://blog.weareopen.coop/badge-wiki-barn-raising-save-the-date-964993885ef0

29 https://badge.wiki/wiki/Main_Page

developing the Web Literacy Map.[30] Community calls are a great 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 most of your community members. An inclusive technique to scheduling is to run your call three times, once for Europe/Ameri-

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

cas, 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 personal 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[31] —or get in touch![32]

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

32 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[33] or via our newsletter[34] and on social media by following @mozillafestival. You'll also find recordings and notes from our calls on the Movement-Building from Home playlist.[35]

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.

33 https://mozillafestival.org/slack

34 https://mozillafestival.org/newsletter

35 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 get overlooked 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 (see Appendix) 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.[36] 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:

36 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.

Building teams we want to be on (regardless of where they're located)

Allison Matlack

I am a relatively new manager—just two years in, at the time of this writing—but I've worked with leadership teams in a support and advisory capacity for much longer. My image of "good" management is a composite of the management attitudes and behaviors that have positively impacted my own career.

In my experience, the formula for a person's best career includes a partnership between an advocate manager and a highly motivated individual contributor.[37] Both parties should be working together toward shared objectives. This requires managers to shift from a mindset of "*you* own *your* success" to one of "*we* own *our* success."

I've given talks on this topic a few times, and those talks culminate with my favorite example of this kind of advocacy in action: the time my manager had to find a chair for me so I could join the table and pitch a new program at our department fiscal-year planning session. He not only opened the door to opportunity by inviting me to the meeting, but also reinforced my value by ensuring I had an equal place to sit alongside other leaders—he literally gave me a seat at the table.

I'm sharing all of this so you have a good understanding of my state of mind when I went into people management. I knew *exactly* what to do to manage according to open values and principles: set proper context, invest time in skill building, open

37 https://opensource.com/open-organization/17/9/own-your-open-career

doors, remove blockers, serve as an advocate, and empower my team.[38]

So imagine my surprise when I became a manager and realized that this stuff is *not* easy to put into practice—especially when you're managing a distributed team.

If you're new to open management, new to managing a distributed team, or have some experience but are looking for a refresher, here are some pragmatic steps you can take toward fostering a healthy team that thinks and acts openly, regardless of where they're physically located.

Step 1: Establish trust through transparency

Not long after I officially became a people manager, I was wondering if I had made a *terrible* decision. Thankfully, it turned out that I was wrong.

I'd just forgotten the most important step any beginning manager needs to take: establish a solid foundation of trust.

Trust is important in any relationship. But when you're working on a distributed team, intentionally creating space for activities that build trust is essential. My current team spans geographies, from India to the Czech Republic, Germany, Australia, and the United States. This distribution ensures we have some diversity of thought, but it makes getting to know one another on a personal level much more difficult (we can't just opportunistically engage in the kinds of activities that create shared experiences, like coffee or lunch breaks, stopping by each other's desks for informal conversation, attending company meetings together, and so on).

Without intentionally creating space for these types of activities and being explicit about both your values and what you expect from other members of the team, you're leaving open the door to misperceptions. Here's an example: When I first became a manager, I was faced with a situation where I thought I had a problem employee. Despite everything I thought I knew, I was failing to

38 https://opensource.com/open-organization/18/10/understanding-engagement-empowerment

identify and repair the disconnect. (I remember once, in an email, when this person even referenced the fact that I'd given talks on this stuff!) I had no idea what to do because the behavior I perceived didn't align to my expectations or assumptions about this person and their work. Among the advice I got was the suggestion to skip one-on-one meetings if I didn't feel up to them—preserve my energy for more positive things. But that seemed to make the problem worse, to increase the fear and uncertainty we both felt because without a shared office space, these meetings became some of the only deliberate moments we had for real conversation.

Things didn't turn around until I gave up fighting. I took off my armor, threw down my defensive weapons, and met this person in the middle.

"You are right," I said, "I do give talks on this stuff, and I swear I am trying everything I coach everyone else to do. I can't tell if I'm a horrible manager, if there's a skills gap somewhere, or if you're in the wrong role and ready for something else." I looked this person in the eyes and I said what I honestly felt: "I care about you. And I care about this team."

And I can tell you that both of us continuing to show up, allowing ourselves to be vulnerable, and displaying genuine care flipped some switch. We were practicing the best and hardest kind of radical candor, and we cleared up misperceptions on both ends.

Much of the behavior I perceived as troublesome was a natural consequence of our environment, which I just needed to better understand. I learned that this person wasn't a problem employee, just a person who felt unsafe and unsure. And this person learned they didn't need to be afraid of being honest, that I wasn't out to get them—just a new manager genuinely trying her best and still stumbling.

The foundation of trust we built that day has gotten even stronger over time. We both feel safe, respected, and supported. And now this management thing isn't so hard because I have a partner who is equally as invested in our mutual success. What I learned from this experience was the importance of establishing

trust out of the gate. So when I recently began onboarding our newest team member, I shared this story and set the intention of having a clear conversation about what we both expected and needed from each other in this manager/associate relationship. If you haven't ever had this conversion with the people on your team, or if it's been a long time since you have, schedule time to check in.

Step 2: Co-create a team charter

I hear many stories from people about how terrible their managers are. But I'm not convinced there's a cavalcade of heartless, cruel managers out there who spend their days thinking of ways to make their associates' lives more difficult. On the contrary: When I start digging into the reasons why a manager/associate relationship isn't healthy, it is almost always because expectations are unclear or misaligned. An associate cannot feel like they are contributing value or be seen as a high performer if they are not meeting expectations, and they cannot meet expectations that are unclear (or worse, unstated).

When forming a new team, stepping in to lead an existing team, combining different functions under one department, or bringing on new team members, the first thing you should do after establishing a foundation of trust is to co-create a team charter. Your charter should answer why your team exists (purpose statement), the ideal end-state you are driving toward (vision statement), how you plan to get to that end state (mission statement), what tactics you will employ to accomplish your mission (objectives, strategies), and how you will know you are successful (key performance indicators [KPIs] or other success measures). In a remote, distributed environment, you and your team can develop all this virtually, using shared documents and slide-sharing tools.

Coordination among remote team members is much easier when those members share a common vision because they can refer to their charter in the absence of more frequent day-to-day reinforcements. Members of teams with a charter can set individ-

ual performance and development goals much more easily—and ensure they're aligned with the team's higher purpose. People readily know how to accept and prioritize work (i.e., if a project is not aligned to the team objectives, then it's not something they should take on, unless it's a passion project outside their core responsibilities). Everyone knows what's expected of them: Help the team accomplish its mission through the defined objectives.

Don't forget that as manager, you should have a general idea of what you'd *like* to see for each of these categories, and providing drafts of each of these statements can also be helpful (sometimes, team members prefer to respond rather than invent—tell you what's missing, indicate whether it resonates, discuss whether the language makes sense and accurately describes their vision for individual roles, etc.). It's your job to use your understanding of the larger business context to guide the conversation about how the team fits in the larger organization, but be open to the team's suggestions—they might surprise you by showing you something you've missed!

To give you an idea of what all of these statements look like, I've included examples from my own team (internal communications for an enterprise software Products and Technologies division).

Why are you here? (purpose statement)

A team's purpose statement should answer the questions, "Why are you here? What makes you different from every other team in the organization?" Clarifying your team's unique role in the organization can help you set bounding lines for responsibilities and define how you partner with other teams.

An exercise that can help you arrive at this statement (especially if you have multiple functions within your team) is to provide a shared document and have every member of the team write up to three sentences that describe what they do. Chances are high that you will see a few words or themes repeated throughout each of the descriptions, and that can help you understand how the team

defines their purpose—and you can refer back to these statements to help with your vision and mission statements, too.

My team's current purpose statement: "Creating the context associates need to do their best work."

What are you driving toward? (vision statement)

A vision statement should describe the ideal end-state your team is driving toward. In other words, what would it take for your team to be ultimately so successful that your job would be considered done and your team would no longer be needed? What do you want to achieve?

My team's current vision statement: "Achieve a sense of community within our division and a deep understanding of how the product strategy is relevant to each of us."

How are you going to get there? (mission statement)

Now that you understand why you are here and where you are going, you need to define how you are going to get there. A mission statement describes how you are going to achieve your vision. What kinds of things will your team do to ensure your vision becomes a reality?

My team's current mission statement: "Create and curate meaningful content, events, and experiences that connect our division's leadership and associates to our open culture, shared mission, and each other."

What are your tactics? (objectives, strategies)

Now we're into the nitty-gritty. What does your team hope to accomplish (objectives) in support of your mission, and how do you plan to meet those objectives (strategies)?

When team members begin creating their own individual performance and development plans, and when they're deciding what projects to take on, they will look to the objectives for guidance. If a project or task is not aligned to the team objectives, then it's not something they should be focusing on as part of their core responsibilities. And if the objectives are clear, you can avoid the problem of misaligned expectations.

It's a good idea to keep the number of objectives small (no more than three or four), then the team can come up with three or four strategies that define ways you'll accomplish each of those objectives.

My team currently has three objectives. As an example, here is one: "Partner with division executives to better connect with the division globally so every associate understands our strategy and how their work aligns and matters."

And that objective's accompanying strategies:

- "Empower division associates with the information and resources they need to understand how their roles fit into the overall strategy,
- Create opportunities for division associates and executives to interact, and
- Provide a platform for division leaders to share their perspectives and demonstrate expertise."

How will you know if you're successful? (success measures)

To help define key performance indicators (KPIs), I like to ask the team the question, "What data, if we had it, would change our behavior?" But be warned that you might not always find a clear answer to that question. For example, we're an internal communications team, so we can try to measure things like publication subscriber numbers and click-through rates—but trends in those numbers are only useful within a larger context (e.g., did more people access content because they were curious about something important going on in the organization, or did we do more promotion than usual?).

Rather than share an example of some KPIs here, I'll just note that sometimes the conversation is more important than the numbers themselves. My team decided to implement a quarterly review process where we'd review our metrics, but after our first trial run, we moved the KPIs to the very end and spent the majority of our time discussing what goals we'd set for the previous quarter, what we actually accomplished, what challenges we encountered, and what we wanted to focus on for the upcoming quarter.

Step 3: Keep each other in the loop

Scrum teams have daily standup meetings, but the rest of us can have difficulty finding ways to keep each other informed of what we're working on and where we might need some help—especially if the team is distributed. Having team meetings on a regular, reliable cadence is critical, and it's equally as important to find an *asynchronous* way to provide updates so you can use those team meetings for discussions about issues that need attention, rather than sharing status.

One of my favorite things that my team has suggested implementing so far has been a weekly high-level update. We use a Trello board on which everyone has an individual card, but the tool you use here doesn't matter; I've heard others use tools like Slack, Google Docs, or email for these updates, so go with whatever makes your team most comfortable with rather than trying to introduce a new tool. I follow our Trello board, so I get the updates straight to my inbox.

Once a week, each person makes a list of "things I'm happy with" and "things that could have gone better," in order to share highlights and lowlights (rather than comprehensive weekly task lists) with each other. These updates serve several purposes. First, everyone on my team has slightly different responsibilities and works on different things, so these updates provide a way for us to keep each other informed about what we're doing ("Met with the intranet team to clarify navigation structure"), seek help removing blockers ("Had trouble figuring out how to format the newsletter"), and identify areas for collaboration ("Started work on our team intranet space")—all outside of the hour-long weekly team meeting.

These updates also provide a vehicle for us to share accomplishments and things we're proud of ("Proud of the team for being recognized in the department meeting"), as well as frustrations on both personal and professional levels—I have found that the more open I am in my updates about what's happening in my life ("Had trouble focusing this week due to the US presidential election"), the more open my team is. And if anything stands out (positive or

negative), I can bring it up in our one-on-one meetings to offer congratulations or support.

And finally, these updates provide a historical record of selected accomplishments on which we can all reflect. I've found that we all refer to the updates to help build our quarterly reviews, as well as individual performance and development plans.

Inspired by my team's suggestion to provide individual updates, I created Trello cards to share notes from each of my standing meetings (extended leadership meetings, working groups, cross-functional groups, etc.) so that I can be more transparent and make information available to my team as they want it, without clogging their inboxes or overloading them with information that doesn't interest them.

And while on the subject of team meetings, I must say I cannot understate the importance of regular one-on-one meetings with all of your team members, especially when they're remote. The people on your team need to know you are there for them, and a regular touch-base is essential. These meetings are your opportunity to check in with your team on a human level. Let them direct the conversation—sometimes you'll end up talking about anything other than work, and that's okay—and don't forget to ask, "How are you doing, really?" (and mean it).

If you have too many direct reports to manage weekly or bi-weekly meetings with all of them, then start a conversation with your own management chain about how to work toward a more sustainable and supportive organizational structure.

Conclusion

Just two years into my career managing a distributed team, I can already say that my favorite part of management is helping create a team I want to be on. What I've learned is that once you've established trust, co-created a team charter, and found a way to keep each other in the loop, something magical happens. I hope you will try it and see.

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 a community architect in the Open Source Program Office at Red Hat. He assists the Open Organization project and community, manages the Open Organization section of Opensource.com, and helps maintain the *Open Organization* book series.

Ben Cotton is a meteorologist by training, but weather makes a great hobby. Ben works as a the Fedora Program Manager at Red Hat. He's an Open Organization Ambassador, an Opensource.com Correspondent, and a general opiner on technology. Ben produces hot takes on Twitter as @FunnelFiasco.

Isabel Drost-Fromm is a member of the Apache Software Foundation, a co-founder and board member of the InnerSource Commons Foundation, and a member of the inaugural Open Source and Intellectual Property Advisory Group of the United Nations Technology Innovation Labs (UNTIL). She's interested in all things search and text mining, and has experience in open source project governance and open collaboration. She's currently working at Europace AG as Open Source Strategist.

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 Coop, 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. He is also a former leader of open source program offices at Motorola, Red Hat, Samsung, and Autodesk. Find him on Twitter as @guyma.

Allison Matlack is a Returned Peace Corps Volunteer turned communications lead at Red Hat who is known for her enthusiastic speaking style and passion for helping other leaders inspire their teams. She is a comms specialist with a change-management style steeped in the tradition of the Open Decision Framework and is honored to have been an Open Organization Ambassador since 2016. Allison loves helping others find ways to put open principles into practice, so reach out on Twitter at @allisonsm7.

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.

Jimmy Sjölund is a Lean and Agile coach at Telia Company, focusing on organisation transformation and team excellence while exploring agile and lean workflows. He is a visualisation enthusiast and an Open Organization Ambassador.

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.

Sim Zacks is a DevOps/CI Architect with more than 20 years of experience across the technology stack, including design and development of software and databases as well as DevOps, CI, and system & network administration. He has experience with business processes development, team management, consensus building, mentoring and training.

Appendix

The Open Organization Definition

The Open Organization Ambassadors

Preamble

Openness is becoming increasingly central to the ways groups and teams of all sizes are working together to achieve shared goals. And today, the most forward-thinking organizations—whatever their missions—are embracing openness as a necessary orientation toward success. They've seen that openness can lead to:

- **GREATER AGILITY,** as members are more capable of working toward goals in unison and with shared vision;
- **FASTER INNOVATION,** as ideas from both inside and outside the organization receive more equitable consideration and rapid experimentation, and;
- **INCREASED ENGAGEMENT,** as members clearly see connections between their particular activities and an organization's overarching values, mission, and spirit.

But openness is fluid. Openness is multifaceted. Openness is contested.

While every organization is different—and therefore every example of an open organization is unique—we believe these five characteristics serve as the basic conditions for openness in most contexts:

- Transparency
- Inclusivity
- Adaptability
- Collaboration
- Community

Characteristics of an Open Organization

Open organizations take many shapes. Their sizes, compositions, and missions vary. But the following five characteristics are the hallmarks of any open organization.

In practice, every open organization likely exemplifies each one of these characteristics differently, and to a greater or lesser extent. Moreover, some organizations that don't consider themselves open organizations might nevertheless embrace a few of them. But truly open organizations embody them all—and they connect them in powerful and productive ways.

That fact makes explaining any one of the characteristics difficult without reference to the others.

Transparency

In open organizations, transparency reigns. As much as possible (and advisable) under applicable laws, open organizations work to make their data and other materials easily accessible to both internal and external participants; they are open for any member to review them when necessary (see also *inclusivity*). Decisions are transparent to the extent that everyone affected by them understands the processes and arguments that led to them; they are open to assessment (see also *collaboration*). Work is transparent to the extent that anyone can monitor and assess a project's progress throughout its development; it is open to observation and potential revision if necessary (see also *adaptability*).

In open organizations, transaprency looks like:

- Everyone working on a project or initiative has access to all pertinent materials by default.
- People willingly disclose their work, invite participation on projects before those projects are complete and/or "final," and respond positively to request for additional details.
- People affected by decisions can access and review the processes and arguments that lead to those decisions, and they can comment on and respond to them.

- Leaders encourage others to tell stories about both their failures and their successes without fear of repercussion; associates are forthcoming about both.
- People value both success and failures for the lessons they provide.
- Goals are public and explicit, and people working on projects clearly indicate roles and responsibilities to enhance accountability.

Inclusivity

Open organizations are inclusive. They not only welcome diverse points of view but also implement specific mechanisms for inviting multiple perspectives into dialog wherever and whenever possible. Interested parties and newcomers can begin assisting the organization without seeking express permission from each of its stakeholders (see also *collaboration*). Rules and protocols for participation are clear (see also *transparency*) and operate according to vetted and common standards.

In open organizations, inclusivity looks like:

- Technical channels and social norms for encouraging diverse points of view are well-established and obvious.
- Protocols and procedures for participation are clear, widely available, and acknowledged, allowing for constructive inclusion of diverse perspectives.
- The organization features multiple channels and/or methods for receiving feedback in order to accommodate people's preferences.
- Leaders regularly assess and respond to feedback they receive, and cultivate a culture that encourages frequent dialog regarding this feedback.
- Leaders are conscious of voices not present in dialog and actively seek to include or incorporate them.
- People feel a duty to voice opinions on issues relevant to their work or about which they are passionate.

- People work transparently and share materials via common standards and/or agreed-upon platforms that do not prevent others from accessing or modifying them.

Adaptability

Open organizations are flexible and resilient organizations. Organizational policies and technical apparatuses ensure that both positive and negative feedback loops have a genuine and material effect on organizational operation; participants can control and potentially alter the conditions under which they work. They report frequently and thoroughly on the outcomes of their endeavors (see also *transparency*) and suggest adjustments to collective action based on assessments of these outcomes. In this way, open organizations are fundamentally oriented toward continuous engagement and learning.

In open organizations, adaptability looks like:

- Feedback mechanisms are accessible both to members of the organization and to outside members, who can offer suggestions.
- Feedback mechanisms allow and encourage peers to assist one another without managerial oversight, if necessary.
- Leaders work to ensure that feedback loops genuinely and materially impact the ways people in the organization operate.
- Processes for collective problem solving, collaborative decision making, and continuous learning are in place, and the organization rewards both personal and team learning to reinforce a growth mindset.
- People tend to understand the context for the changes they're making or experiencing.
- People are not afraid to make mistakes, yet projects and teams are comfortable adapting their pre-existing work to project-specific contexts in order to avoid repeated failures.

Collaboration

Work in an open organization involves multiple parties by default. Participants believe that joint work produces better (more effective, more sustainable) outcomes, and specifically seek to involve others in their efforts (see also *inclusivity*). Products of work in open organizations afford additional enhancement and revision, even by those not affiliated with the organization (see also, *adaptability*).

In open organizations, collaboration looks like:

- People tend to believe that working together produces better results.
- People tend to begin work collaboratively, rather than "add collaboration" after they've each completed individual components of work.
- People tend to engage partners outside their immediate teams when undertaking new projects.
- Work produced collaboratively is easily available internally for others to build upon.
- Work produced collaboratively is available externally for creators outside the organization to use in potentially unforeseen ways.
- People can discover, provide feedback on, and join work in progress easily—and are welcomed to do so.

Community

Open organizations are communal. Shared values and purpose guide participation in open organizations, and these values—more so than arbitrary geographical locations or hierarchical positions—help determine the organization's boundaries and conditions of participation. Core values are clear, but also subject to continual revision and critique, and are instrumental in defining conditions for an organization's success or failure (see also *adaptability*).

In open organizations, community looks like:

- Shared values and principles that inform decision-making and assessment processes are clear and obvious to members.
- People feel equipped and empowered to make meaningful contributions to collaborative work.
- Leaders mentor others and demonstrate strong accountability to the group by modeling shared values and principles.
- People have a common language and work together to ensure that ideas do not get "lost in translation," and they are comfortable sharing their knowledge and stories to further the group's work.

Revision History

Version 2.0
April 2017
github.com/open-organization/open-org-definition

Additional Resources

Learn more

Explore additional resources online,[39] then follow the project on Twitter[40] and LinkedIn.[41]

Make your contribution

The Open Organization book series is open source and always accepting new contributions. Share your knowledge and your experience by joining the project on GitHub.[42]

Join the community

Are you passionate about using open source ideas to enhance organizational life? Connect with like-minded practitioners at the Open Organization community hub.[43]

---

39 http://www.theopenorganization.org/

40 https://www.twitter.com/openorgproject

41 https://www.linkedin.com/company/the-open-organization

42 https://www.github.com/open-organization

43 https://www.theopenorganization.community/