Harvard Business Review

REPRINT H05UH7
PUBLISHED ON HBR.ORG
SEPTEMBER 09, 2020

ARTICLE
COLLABORATION

Remote Work Doesn’t Have to Mean All-Day Video Calls

by Marco Minervini, Darren Murph and Phanish Puranam

Harvard Business Review

COLLABORATION

Remote Work Doesn't Have to Mean All-Day Video Calls

by Marco Minervini, Darren Murph and Phanish Puranam

SEPTEMBER 09, 2020

EDWIN REMSBERG/GETTY IMAGES

The Covid-19 crisis has distanced people from the workplace, and employers have generally, if sometimes reluctantly, accepted that people can work effectively from home. As if to compensate for this distancing and keep the workplace alive in a virtual sense, employers have also encouraged people to stick closely to the conventional workday. The message is that working from home is fine and can even be very efficient — as long as people join video calls along with everyone else all through the day.

But employees often struggle with the "workday" when working from home, because many have to deal with the competing requests coming from their family, also housebound. So how effective really is working from home if everyone is still working to the clock? Is it possible to ditch the clock?

The answer seems to be that it is. Since before the pandemic we've been studying the remote work practices of the tech company GitLab to explore what it might look like if companies to break their employees' chronological chains as well as their ties to the physical workplace.

The challenge for GitLab

From its foundation in 2014, GitLab has maintained an all-remote staff that now comprises more than 1,300 employees spread across over 65 countries. The "git" way of working uses tools that let employees work on ongoing projects wherever they are in the world and at their preferred time. The idea is that because it's always "9 to 5" somewhere on the planet, work can continue around the clock, increasing aggregate productivity. That sounds good, but a workforce staggered in both time and space presents unique coordination challenges with wide-ranging organizational implications.

The most natural way to distribute work across locations is to make it **modular** and independent, so that there is little need for direct coordination — workers can be effectively without knowing how their colleagues are progressing. This is why distributed work can be so effective for call centers and in patents evaluation. But this approach has its limits in development and innovation related activities, where the interdependencies between components of work are not always easy to see ahead of time.

For this kind of complex work, co-location with **ongoing communication** is often a better approach because it offers two virtues: synchronicity and media richness. The time lag in the interaction between two or more individuals is almost zero when they are co-located, and, although the content of the conversation may be the same in both face-to-face and in virtual environments, the technology may not be fully able to convey soft social and background contextual cues — how easy is it to sense other people's reactions in a group zoom meeting?

All this implies that simply attempting to replicate online (through video or voice chat) what happened naturally in co-located settings is unlikely to be a winning or complete strategy. Yet this approach of "seeing the face" is the one that people seem to default to when forced to work remotely, as our survey of remote working practices in the immediate aftermath of lockdowns around the world has revealed.

Tacit coordination

There is a way through this dilemma. Our earlier research on offshoring of software development showed that drawing on **tacit coordination** mechanisms, such as a shared understanding of work norms and context, allows for coordination without direct communication.

Coordination in this case happens through the observation of the action of other employees and being able to predict what they will do and need based on shared norms. It can occur either **synchronously** (where, for instance, two people might work on the same Google doc during the same time period), **or asynchronously** (when people make clear hand-offs of the document, and do not work on it when the other is).

Software development organizations often opt for this solution and tend to rely extensively on shared repositories and document authoring tools, with systems for coordinating contributions (e.g., continuous integration and version control tools). But GitLab is quite unique in the for-profit sector in how extensively it relies on this third path not only for its coding but for how the organization itself functions. It leans particularly on asynchronous working because its employees are distributed across multiple time zones. As a result, although the company does use videoconferencing, almost no employee ever faces a day full of video meetings.

How it works at GitLab

At the heart of the engineering work that drives GitLab's product development is the "git" workflow process invented by Linux founder Linus Torvalds. In this process, a programmer making a contribution to a code "forks" (copies) the code, so that it is not blocked to other users, works on it, and then makes a "merge request" to have the edited version replace the original, and this new version becomes available for other contributions.

The process combines the possibility of distributed asynchronous work with a structure that checks for potential coordination failures and ensures clarity on decision rights. Completely electronic (which makes remote work feasible) and fully documented, it has become an important framework for distributed software development in both for-profit and open source contexts.

GitLab has taken the git a step further, applying it also to managerial work that involves ambiguity and uncertainty. For instance, GitLab's chief marketer recently outlined a vision for integrating video into the company's year-ahead strategy. He requested asynchronous feedback from across the company within a fixed time window, and then scheduled a single synchronous meeting to agree on a final version of the vision. This vision triggered asynchronously input changes from multiple contributors to the company's handbook pages relating to marketing objectives and key results that were merged on completion.

GitLab's high degree of reliance on asynchronous working is made possible by respecting the following three rules right down to the task level:

1. Separate responsibility for doing the task from the responsibility for declaring it done.

In co-located settings, where employees are in the same office, easy communication and social cues allow them to efficiently resolve ambiguities and manage conflict around work responsibilities and remits. In remote settings, however, this can be difficult. In GitLab, therefore, every task is expected to have a Directly Responsible Individual (DRI), who is responsible for the completion of the task and has freedom in *how* it should be performed.

The DRI, however, does not get to decide whether the task has been completed. That function is the responsibility of a "Maintainer," who has the authority to accept or reject the DRI's merge requests. Clarity on these roles for every task helps reduce confusions and delays and enables multiple DRIs to work in parallel in any way they want on different parts of a code by making local copies ("forking"). It is the Maintainer's role to avoid unnecessary changes and maintain consistency in the working version of the document or code.

In a non-software context, say in developing the GitLab handbook page on expenses policies, individual DRIs, who could be anyone in the company, would write specific policies in any way they choose, and their contributions would be accepted or rejected by the CFO acting in the capacity of Maintainer, who could also offer feedback (but not direction) to the DRIs. Once live, the merged page serves as the single source of truth on expenses policies unless or until someone else makes a new proposal. Once more, the Maintainer would approve, reject, or offer feedback on the new proposal. In contexts like this, we would expect people in traditional management positions to serve as Maintainers.

2. Respect the "minimum viable change" principle.

When coordination is asynchronous, there is a risk that coordination failures may go undetected for too long – for instance, two individuals may be working in parallel on the same problem, making one of their efforts redundant, or one person may be making changes that that are incompatible with the efforts of another. To minimize this risk, employees are urged to submit the minimum viable change — an early stage, imperfect version of their suggested changes to code or documents. This makes it more likely that people will pick up on whether work is incompatible or being duplicated. Obviously, a policy of minimum viable changes should come with a "no shame" policy on delivering a temporarily imperfect output. In remote settings, the value of knowing what the other is doing as soon as possible is greater than getting the perfect product.

3. Always communicate publicly.

As GitLab team members are prone to say, "we do not send internal email here." Instead, employees post all questions and share all information on the Slack channels of their teams, and later the team leaders decide what information needs to be permanently visible to others. If so, it gets stored in a place available to everyone in the company, in an "issue" document or on a page in the company's online handbook, which is accessible to anyone, in or outside the company. This rule means that people don't run the risk of duplicating, or even inadvertently destroying the work of their colleagues. Managers devote a lot of time to curating the information generated through the work of employees they supervise and are expected to know better than others what information may be either broadly needed by a future team or that would be useful for people outside the company.

Recognizing the limits of the GitLab approach

However well implemented, asynchronous remote working of this kind cannot supply much in the way of social interaction. That's a major failing, because social interaction is not only a source of pleasure and motivation for most, it is also where the "random encounters," the serendipitous exchanges by the coffee machines and lift lobbies, create opportunities for ideas and information to flow and recombine.

To minimize this limitation, GitLab provides occasions for non-task related interaction. Each day, team members may attend one of three optional social calls — staggered to be inclusive of time zones. The calls consist of groups of 8-10 people in a video chatroom, where they are free to discuss whatever they want (GitLab provides a daily starting question as icebreaker in case needed, such as: "What did you do over the weekend?" or "Where is the coolest place you ever traveled and why?").

In addition, GitLab has social slack groups: thematic chat rooms that employees with similar interests can participate in (such as: #cat, #dogs, #cooking, #mental_health_aware, #daily_gratitude, #gaming) and a #donut_be_strangers channel that allows strangers that have a mutual interest to have a coffee chat to get together.

Of course, GitLab managers are under no illusion that these groups substitute perfectly for the kinds of rich social interactions outside work that people find rewarding. But they do help to keep employees connected, and, at a time when many employees have been working under confinement rules, this has proved very helpful in sustaining morale.

***

Working from home in an effective way goes beyond just giving employees a laptop and a Zoom account. It encompasses practices intended to compensate or avoid the core limitations of working remotely, as well as fully leverage the flexibility that remote can offer — working not only from anywhere but at any desired time. We have focused on GitLab because it not only has extensive experience in remote working but also because it pursues an unusual mode of solving the intrinsic challenges of remote work. While some of GitLab's core processes (like its long, remote onboarding process for new hires) and advantages (like the possibility of hiring across the world) cannot be fully reproduced in the short run in companies that will be just temporarily remote, there are others that any company can easily implement.

Marco Minervini is Post-Doctoral Fellow at INSEAD, a global business school with campuses in Abu Dhabi, France, and Singapore

Darren Murph is Head of Remote at GitLab, and author of Gitlab's "Remote Playbook" and "Living the Remote Dream"

Phanish Puranam is the Roland Berger Chair Professor of Strategy & Organization Design at INSEAD.