Future wants for remote software engineering

Vanessa Sochat, *Research Software Engineer, Stanford University Research Computing Center*

As humans, we tend to only notice those experiences or qualities that are not comfortable.

In a working environment, we are less likely to notice when we feel comfortable and are relatively happy, and are more likely to notice when things come to naught [14]. The same is true for remote work, and specifically, software engineering, which requires a delicate balance of collaboration, concentration, and care. In these quickly changing times [19] when remote work is becoming more common out of necessity, we are quickly learning that there are benefits that range from increased productivity to decreased corporate expenses [15]. While these changes are immensely good, there are significant challenges that are placed on the shoulders of workers [22]. There are specific features of in-person interaction that are still poorly addressed in the increasingly remote world. In this article, we point out these challenges, and discuss potential futures that might address them.

Responsibility for Routine

Routine at an office

When working in an office, the bulk of needs for a software engineer are largely met by an employer. For snacks or coffee there might be a break room or micro-kitchen [21]. For meal time, if there aren't in-house cafeterias, many companies are centrally located amidst a good selection of eateries. This routine to get coffee or lunch also carries with it social benefits. The lunch hour or break time is an expected time to enjoy chatting with colleagues, or even just being around other people. Some workers enjoy getting ready in the morning, and looking nice for work - an incentive that might not be as easily summoned when working from home. Even the daily schedule is controlled by the routine of coming into an office - there are windows of clear starting and ending times. The commute itself might be a time to listen to the radio, or in the case of using public transportation, catch up on work or read a book. Although these options aren't always perfectly liked by employees, they set a clear structure to follow that can be hugely helpful.

Routine at the home office

This corporate structure can now be contrasted with remote work. Although there might be virtual meetings, there are no longer any perks that come with them. There is no break room with coffee ready to go, nor a cafeteria where the developer is likely to find friends for lunch. There is no well-defined delineation between the start and end of a day, so work tends to meld with everyday life. It's clear from these basic points that the burden of responsibility falls on the shoulders of the remote worker. It's up to them to find a routine that works, and to manage the huge number of new distractions present at home.

A shared responsibility

For the future of remote work to be more tolerable, we would want some of this burden of responsibility to be taken off of the shoulders of the workers. It might be the case that remote work comes with regular care packages, or subscriptions to resources such as food or services. A remote developer should be able to choose from a selection of pre-determined meal delivery and coffee machines, and have some allowance for home office equipment. A company might also consider creating more, smaller satellite offices to house fewer people and provide these same resources. If the employer is truly benefiting by way of increased productivity and lower costs of operation, this should be feasible to do. The fewer things that the remote worker needs to worry about, namely things that were not worried about at the office, the better.

Virtual Collaboration

Whiteboards

One of the most fun, and fulfilling experiences for a research software engineer is to hash out a problem or system design on a white board. For many of us coming from academia, this was a natural way to work together on a problem. While basic tools exist for writing on a shared document [5], ideally with some kind of tablet with a stylus, these tools are a far cry from the experience of drawing on a large, shared board with other developers. What is needed, and to be developed on a reasonable level for an average consumer, is a virtual whiteboard that is akin to a real whiteboard, but also projects other remote workers' scribbles, in real time, and ideally with integrated audio and visual to support conversation and awareness of where your collaborator is currently inspecting.

Virtual Workplaces

While many online games and interfaces exist that can afford interaction [4, 12, 16], it would be hugely beneficial to have a workplace-scoped environment that might take a set of currently only virtual interactions (chatting on Slack, having meetings, socializing with colleagues remotely) and giving them context and volume in a virtual space. These virtual environments would be a direct way to address some of the mental health challenges associated with remote work, namely loneliness and isolation [11]. For example, a user would log in to their office, perhaps have a small social interaction or routine to start the day (having coffee with a colleague or playing a game), and then head into a virtual office, both customized for the particular institution and team, and then further customized by the individual on the level of his or her office. Just like a real working space, there might be meeting rooms to go into, a board to post announcements, and a cafeteria to go sit if you want to have lunch and socialize. In the same way that an individual might have previously enjoyed going into the office, that same individual should have positive rewards when going into a virtual office.

Institutional Collaboration

It's currently way too challenging for different organizations to easily work together. While national resources exist for compute (e.g., XSEDE [6]), and organizations for software or sustainability [8, 10, 20] it still is the case that collaboration is challenging – only happening by way of extensive paperwork, legal documents, and often finding loopholes in an institutional payment structure. While academic centers, national labs, and other research institutes are trying to solve similar problems, it's often the case that they work in isolation, using software engineers and resources limited to their institution. Akin to open source work, an ideal future would allow for these projects to exist on the same level that open source software does, meaning that there is a transparent network of research software engineers and resources that can be called upon to work together on a project. A national fabric of these research software engineers would foster better communication, and consequently more collaboration and derivation of more informed solutions to shared challenges. This openness should extend to collaborations with industry, meaning that an industry partner that wants to support a project for science can easily do so without many months of paperwork that likely disincentivize the collaboration from happening again. It is only when we bring down these barriers to inter-institutional collaboration that come in the way of licensing and paperwork that we will be able to achieve a truly efficient and streamlined remote working culture.

Learning and Knowledge Transfer

On Demand Training

When we think of training, we often think of classroom settings, online courses, or supervised exercises. We can browse well-known sites like the Khan Academy [7] or Coursera [3] to find predefined sets of knowledge that might help us in our daily work or serve as a credential to be assessed for a new role. While interactive lessons might have served us well in earlier life, they arguably might not be the best approach for a remote work force that requires mostly asynchronous learning. Creating and maintaining interactive lessons would also require substantial resources. Even if this effort is given, it is still the case that these courses are not as effective as in-person learning [18].

The problem at hand is not only about the delivery method, but also that we don't fully understand what is useful to know. The high level topics that institutions would tend to offer as courses aren't necessarily the topics that would be most beneficial to be taught. Further, if an employer asked their team of research software engineers what they might want or benefit from learning in advance, it's not clear that the engineers could easily answer this question.

Thus, what needs to be worked on first is not a new method, environment for learning, or course topic, but rather tested methods for better understanding how research software engineers learn, and in those cases, what knowledge has proved to be most useful. This might be achieved by asking software engineers retroactively, "What things did you learn that you wish would have been more organized? What things were you exposed to that might have been popular but didn't necessarily help you?"

Given that we understand these needs, in a potential future, a research software engineer should be able to sit down with their supervisor and openly discuss goals and desires for learning. The burden does not have to fall entirely on the supervisor – productivity researchers might also study which learning topics are most helpful for different roles for additional guidance. The supervisor can then help to lead the engineer toward projects and experiences that are in line with those goals.

A Learning Marketplace

Better communication between institutions would allow for creation of a clearinghouse for collaboration. This interface between mentors and mentees might even be managed by an unexpected third party that cares about open source development [1]. This marketplace would serve as a hub of connection, one where organizations post projects that are actively looking for help, and individuals go to provide it. The marketplace needs to be large enough to extend across industry and academia, and have incentives set up for all participants. For example, some projects might provide compensation to the individual or organization, and there would be payment streams set up in advance to handle that. Other projects might include an individual as an author of a paper. A manager should then be able to create posts on the clearinghouse that indicate looking for collaborators, and also look for opportunities for the engineers that they represent. This is how learning can optimally happen – through real experiences that are seeking to add real value. Arguably, the best learning happens when we aren't aware of it at all.

On-boarding

In-person on-boarding can be anything from getting access to a shared document, to a day with a welcome breakfast to a week of talks and fun events. This process is generally set up to familiarize an individual with an organization's people, rules, and protocols, and also serves as a welcoming event [13]. With the switch to remote work, much of this interactive experience is no longer possible. However, the need still exists for a new engineer to become familiar with new faces and tools in their work place. While on-boarding is typically understood as one concept or event, there are actually two components that make it up. The first is an institution-level on-boarding where an employee learns about organizational resources, rules, and challenges. Instead of having one intense session at the onset of a new job, a suggested approach would be to create virtual social events that are held on a regular basis. Each session would cover a well scoped topic, having a short presentation followed by discussion. The approach would take the stance that learning and getting acquainted with your institution is not a one time thing, but a work in progress that persists as long as an individual remains at an organization. The second component that is wrapped into the idea of on-boarding is getting familiar with one's team. For this need, an approach might be taken that is similar to doing rotations in graduate school. A new team member should be exposed to many different kinds of projects that the team is working on by working closely with another team member each day. At the end of the experience the new member would know team members more intimately and be able to pick up participation in one or more favorite projects.

Pair Programming

Pair programming, or two individuals sitting at the same computer and working together on a piece of code, is arguably an underutilized resource [9]. Other than providing some social outlet and an opportunity for engineers to learn from one another, in that pair programming forces an engineer to explain their thinking, it helps to strengthen communication skills. Akin to most in-person experiences, pair programming becomes very challenging in a remote context. This problem will be greatly helped by shared, collaborative programming environments [2], and coding spaces that are provided alongside code [17]. The hardest issue in adopting these practices will likely be cultural, as a developer is likely to be self-conscious if writing code in front of others.

Even if we get around these cultural barriers, however, it's still unlikely that an engineer will be incentivized to arrange in advance a pair programming session. Instead, a suggested future would have feature or other development branches of repository code (that more than on individual on a team care about) available to work on in a collaborative coding environment. Developers would visit the environment when they want to work on the code, something that they already would do, and organically stumble on their colleagues. The developers could work in isolation, but just by way of being present at the same time and interested in the same problem, it's very likely that they would chat, either in a text or more comfortable voice channel. This setup would add a live social element to coding that has not previously existed, and likely be beneficial and enjoyed by many developers. Even in the case that a developer does not want to directly interact, it could be helpful to learn by watching what another developer is doing. And of course, if a developer absolutely does not want this interaction, they can still choose to code outside of the collaborative environment.

Discussion

As more individuals start to work remotely, addressing these challenges of communication, organization, and tools will be essential for ensuring that a remote working experience does not fall short of any features that were previously present. We have discussed a small set of future desired, namely:

- Making both employers and employees responsible for establishing daily routine
- Developing consumer-ready, collaborative tools such as whiteboards
- Creating immersive, virtual experiences that mimic a day in the office
- Fostering better inter-institutional collaboration
- Establishing a collaboration clearinghouse to match developers with work
- Transitioning on-boarding from a single event to a regular, fun occurrence
- Using interactive spaces that encourage pair programming

While progress toward these futures might be slow, or for some of the ideas, not possible at all, it's important to have open conversation with developers and managers to better understand the experiences that are missing from remote work that would have previously been available. It's also important to have openness to trying new things. An idea when tested for the first time may not be a booming success, but if it moves a community slightly closer to a new paradigm, it's definitely worth trying.

References

[1] Build software better, together.

[2] Codeshare - share code in real-time with developers in your browser. https://codeshare.io/. Accessed: 2020-6-4.

[3] Coursera — build skills with online courses from top institutions. https://www.coursera.org/. Accessed: 2020-6-4.

[4] Fortnite - chapter 2 — official site — epic games. https://www.epicgames.com/fortnite/en-US/ch2-s2?utm_source=GoogleSearch&utm_medium=Search&utm_campaign=an*Internal_pr*FNBR_ct*Performance_pl*SearchBrand_co*US_cr*exact&utm_id=1698686527&sub_campaign=Fortnite_Exact_EN&utm_content=&utm_term=fortnite&utm_term=fortnite&gclid=EAIaIQobChMI1N7crMjo6QIVBvzjBx2Wug8gEAAYASAAEgJCaPD_BwE. Accessed: 2020-6-4.

[5] Google jamboard: Collaborative digital whiteboard — G suite for education — google for education. https://edu.google.com/products/jamboard/?modal_active=none. Accessed: 2020-6-3.

[6] Home - XSEDE. https://www.xsede.org/. Accessed: 2020-6-3.

[7] Khan academy — free online courses, lessons & practice. https://www.khanacademy.org/. Accessed: 2020-6-4.

[8] The national center for supercomputing applications at the university of illinois at Urbana-Champaign. http://www.ncsa.illinois.edu/. Accessed: 2020-6-3.

[9] On pair programming. https://martinfowler.com/articles/on-pair-programming.html. Accessed: 2020-6-4.

[10] The software sustainability institute — software sustainability institute. https://www.software.ac.uk/. Accessed: 2020-6-3.

[11] State of remote work 2019. https://buffer.com/state-of-remote-work-2019. Accessed: 2020-6-3.

[12] If second life isn't a game, what is it? http://www.nbcnews.com/id/17538999/ns/technology_and_science-games/t/if-second-life-isnt-game-what-it/, March 2007. Accessed: 2020-6-4.

[13] What is employee onboarding process? definition, templates, and best practices. https://hr.toolbox.com/articles/what-is-new-employee-onboarding/, June 2020. Accessed: 2020-6-4.

[14] Kendra Cherry. Why our brains are hardwired to focus on the negative. https://www.verywellmind.com/negative-bias-4589618, April 2019. Accessed: 2020-6-3.

[15] Laurel Farrer. 5 proven benefits of remote work for companies. *Forbes Magazine*, February 2020.

[16] Patrick Klepek. 'world of warcraft' changed video games and wrecked lives. https://www.vice.com/en_us/article/ywaj4w/world-of-warcraft-changed-video-games-and-wrecked-lives, November 2019. Accessed: 2020-6-4.

[17] Frederic Lardinois. GitHub gets a built-in IDE with codespaces, discussion forums and more. *TechCrunch*, May 2020.

[18] Susanna Loeb. How effective is online learning? what the research does and doesn't tell us. https://www.edweek.org/ew/articles/2020/03/23/how-effective-is-online-learning-what-the.html?cmp=SOC-SHR-FB, March 2020. Accessed: 2020-6-4.

[19] Sarah Mervosh, Denise Lu, and Vanessa Swales. See which states and cities have told residents to stay at home. *The New York Times*, March 2020.

[20] US-RSE. US-RSE. https://us-rse.org/. Accessed: 2020-6-3.

[21] Talib Visram. This snack curator for google is one of the most powerful people in food. https://www.fastcompany.com/90369645/google-snack-buyer-food-industry-powerhouse, July 2019. Accessed: 2020-6-3.

[22] Zapier. The 7 biggest remote work challenges (and how to overcome them). https://zapier.com/blog/remote-work-challenges/, March 2017. Accessed: 2020-6-3.