Developer productivity means developer happiness

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

At face value, when we think of developer productivity we might think of effectiveness in time management, communication, and task completion [10].

Although we are drawn to personal workflow or time management tools, and learning secrets to improving our productivity, ironically this quest for the holy grail can sometimes take us off course and be a detriment to our productivity. The problem is that accomplishing tasks or having a filled up schedule does not necessarily equate to productivity. Creating a formulaic working strategy, as was common in the last century, does not either [6, 8, 13]. Productivity is less a quality that can be easily measured [7], controlled, or improved directly with tools, but instead is a human element that manifests from developer happiness.

This article aims to draw focus on this human element. As a developer of scientific software, and one that has transitioned to working remotely before any stay at home orders [12], I have slowly learned to optimize my own productivity by focusing exclusively on well-being. In the following sections, I will summarize what I have learned in the "10 best practices for remote Research Software Engineering,"

The 10 Best Practices for Remote Software Engineering

1. Work on things that you care about

It goes without saying that a core ingredient to happiness, and thus productivity, is working on projects or software that you care about. We might even step back to say that before we discuss any best practice criteria, you should enjoy the practices of software engineering, or participation in a community, enough to warrant having it a part of your daily life. If you aren't sure if you are in the right role, then assess it based on the small tasks that it encompasses. You can break your day down into small events or tasks, and take a mental note about whether or not you enjoyed each one. For example, a software engineer might realize that they enjoy working publicly on open source on GitHub, and writing documentation, but are not so keen on re-answering the same questions on a private user forum. Figuring this out would help the developer to push for projects and responsibilities that are bettered catered to his or her interests. If you can't identify any items, or if all the items identified are considered negative experiences, then it's time to look more critically at your role. There is no list of best practices that you could reasonably follow to make you happy if you aren't in the right line of work, and if you aren't happy, you will have a hard time being regularly productive.

2. Define goals for yourself

As research software engineers (RSEng), we usually don't have a career ladder. Many of us don't even sit within established groups where we might have a manager that is aware of supporting our personal development and growth. Software groups in industry often have training programs, career tracks, and managers that are required to care about personal and professional growth. Although some ideal future might also have this kind of structure for a RSEng, in the present moment we have to take our personal and professional development into our own hands. This means thinking about the kind of work that you want to do, or more generally, the goals you want to accomplish in the short and long term. For example, if you program a lot, your goal might be to learn a new language that would be useful to know. If you are feeling disconnected, your goal might be to pursue interacting with a small set of new communities, or creating structure into your schedule for discussing projects or work with colleagues. If you are involved with research, your goal might be to submit or publish a paper. Shorter term goals might be in the context of a week or month. You might

want to solve a particular challenge, add a feature to your software, or write up an idea that you have. Having these goals is important for your productivity because you can better understand how your work fits into the overall story of you, as a RSEng. If you share your goals with your colleagues or supervisor, they might learn that they are important to you, and be interested to support you or otherwise keep an eye out for you.

3. Define productivity for yourself

If we were in a role that explicitly produced some kind of output (e.g., a farmer might grow corn) then we might easily measure and define productivity [3] as these units of output. However, in context of a rich role that involves not just writing code but also interacting with people and ideas, this simple representation doesn't work very well. There is huge variance in RSEng work, and it would be a daunting task to to quantify or qualify what productivity means globally. However, even though we lack definition, we still are cognizant of what a productive day feels like. A productive day is one that you finish and feel satisfaction that your work was meaningful. Thus, it follows that we might be able to define productivity for ourselves by assessing our daily tasks and deciding if they contribute to that feeling of satisfaction. A RSEng might create a list, and add items to it whenever something feels done or accomplished. These items can either be items of work that are mention-worthy, or softer goals like fostering a collaboration or growing a community. This list can be hugely useful for deriving a better understanding of yourself, and what makes you happy or productive. Although happiness and productivity are not exactly the same thing, they seem to be intimately related [11].

4. Establish routine and environment

Small details about your working environment, or lack of a routine, can hugely throw off your work day, and thus your productivity. You should generally pay attention to the lighting, noise level, and comfort of a work space. If you find yourself distracted by anything, you might consider changing your environment to better support work. For example, without stay at home orders, you might be the kind of person that enjoys the routine of going to work in a coffee shop. You might not have preference for one particular place, but enjoy a routine where you move around between several locations. If you find that you are distracted easily, then you can either remove the distractions, or create rules and rewards to enforce good behavior. For example, you might decide that browsing the internet for pleasure is something that you can look forward to after dinner, and decide to not do it during a work day. You might find that you prefer an environment with another person (remote or in person), or that you like working alone. An important aspect of this routine is creating a clear separation between times when you are working, and times when you are not. This is especially challenging and important for working at home, because any routine of going to and from work is gone, and home becomes two in the same. Finally, a routine that includes more than work, meaning activities that are for your personal well-being (exercise, time with family, relaxation), are essential to consider. The remarkable features of a healthy routine and environment are that they work so well that you don't notice them.

5. Take responsibility for your work

If you are in a role that provides user-support, work on a team, or otherwise have a list of tasks set out for you, there is a tendency to be passive. While you might check off items from a to-do list on a daily basis, this doesn't necessarily indicate that you have taken ownership or responsibility for your work. Having this ownership can be a driving factor for caring more about your work, and thus your productivity. It's a challenging but important skill to learn how to independently recognize and address challenges in your community or space. For example, a challenge might be anything from a small annoyance that your group faces on a regular basis, to a significant computational problem. The more capable you are for deciding something is important to do, creating a plan to do it, and doing it the more you can establish independence, learn new skills, and feel that you have ownership over your work. If you share your ideas or projects with your supervisor or team, they will also see these positive qualities in you, and this can help to establish trust, which is hugely important in a remote working environment. Your supervisor should not require tracking software to keep abreast of your every move because they have confidence in you.

6. Take responsibility for human connection

When newly remote, it's common to place the burden of connection on your team. If your team isn't remote first [9], it's even more unlikely that there are established protocol and best practices for how to make you feel included and happy. Many of us learn this the hard way right at the offset of going remote. If we become disconnected from our immediate or even open source communities, it can feel terribly lonely [14]. This is another situation of needing to identify that there is a problem, namely lack of human connection, and figuring out the kinds of human connection that are meaningful to you. For example, you might attend virtual workshops, conference calls, or look around for regular meetings that you could be a participant in. If you are looking for more one-on-one human interaction, you might see if anyone would be interested in having a virtual coffee a few times a month, or a virtual "happy hour" to involve others. The choice of group is up to you. While it's reasonable to consider your colleagues at work, what you are missing might be connection to friends or family, and so a regularly scheduled family meeting or even meeting online to play a game could be a regular event. In the context of work, you can use the trick of sharing your work to pursue human connection. If you share progress on your projects on social media or Slack, it likely will open lines of communication with others interested in your work. As a remote worker you might live in isolation, but you don't need to work in it.

7. Practice empathetic review

Software engineering requires a lot of technical communication. Whether you are having discussion in a chat window, on video, or a code review, you are undoubtedly going to encounter a lot of different kinds of people, all with different expectations and incentive structures. Empathy is essential. For example, if you encounter a maintainer of a repository that seems curt or mean, you might consider the reponsibility on their shoulders to maintain the software, the number of other issues or pull requests they need to review, and that they are working in free time. The maintainer is likely to be in a stressful role. You might also consider cultural differences in communication - it could be that a succinct response is more common in their community or culture. When you take these factors into account, a response that you might have reacted with that is defensive, confrontational, or otherwise aggressive, might be adjusted to have kindness, and further, be productive to ask how you might help. The more streamlined these communications are in terms of supporting good communication and supportive developers, the better the outcomes of them are, whether that be an issue addressed, a problem solved, or a feature added. By the same line of thought, you want to communicate transparently to allow for empathetic development. This means introducing yourself, describing the issue or change clearly, and making it easy for the other party to reproduce. By setting an example, you will not only influence the others involved in the work at hand, but also set an example for the immediate community are larger community of research software engineers.

8. Have self-compassion

You can do your best to define personal goals, establish routine, and do meaningful work that feels productive, but this doesn't always mean that you finish the day and feel that you were successful. This is where self-compassion [4] is important. Self-compassion can broadly be defined as being mindful and forgiving to yourself. Having self-compassion means looking back on the experience of a day, and not egging yourself for everything that wasn't perfect. It's just a given every day will not feel productive. You might try new routines that don't work for you. Instead of thinking negatively, you might consider that establishing your routine, or figuring out a hard problem, is a work in progress, and there is no real state of failure because you wake up and try again. It helps to try to imagine yourself as another person. If someone else approached you with the same experiences and negative thoughts, how would you comfort them? It's very likely that you would react with compassion, and that's the same compassion you must have for yourself. While you cannot control other people, you can respond with kindness and try to find humor in the situation. However, obviously if someone is in violation of some code of conduct, you should report it, and not absorb it quietly. This is especially true for women and other minorities the work force that might experience subtle inequality or microaggressions [1].

9. Learn to say yes, no, and not anymore

As an open source developer, engaging with projects is akin to going to a candy store. Each one offers a rich set of problems to work on, people to work with, and new languages or technologies to learn.

You might be someone that very easily says "yes" to contributing to a project, or generally providing help. While this is a really great way to grow as a developer and person, there is also a time to say no - when you don't have the bandwidth, don't share the vision of the project, or otherwise have a gut feeling telling you to do so. It's also okay to say "yes" but then scope your contribution to an amount that you have time for. Finally, if you originally say "yes" and change your mind? This is okay too. To maximize your productivity, maintaining awareness about what is on your plate, and when it's time to ramp up or taper down engagement is essential for focusing on the projects that are most valuable and important to you or your community.

10. Choose correct communication channels

When working remotely, there are many ways to communicate with others. In that people have different working environments and schedules, ideally the communication is asynchronous, meaning that you use messaging services like Slack, or issue boards on version control services like GitHub [2]. However, you should be careful to choose the method based on the needs of the communication. Discussion that should be open and linked to code would be better on GitHub issues or Gitter [5] than Slack. A discussion that moves into a document might first do really well using a collaborative document tool like Google Docs, but after some hardening you might want to move it into version control, Sometimes a quick call is more efficient than trying to write out a verbose email. The important detail is that, whether you choose a video call, an email, or a Slack message, your choice of channel is appropriate for the needs of the problem that needs to be discussed, the degree to which the communication should be transparent and open, and your own level of comfort. If you are less comfortable with video or voice chat, you might section off specific time slots or days for it to make it a predictable part of your routine.

Discussion

We've just summarized 10 suggested best practices to optimize developer happiness, and thus developer productivity:

- Work on things that you care about
- Define goals for yourself
- Define productivity for yourself
- Establish routine and environment
- Take responsibility for your work
- Take responsibility for human connection
- Practice empathetic review
- Have self-compassion
- Learn to say yes, no, and not anymore
- Choose correct communication channels

You might have noticed that most of these points come down to identifying a locus of control. The developer that feels in control of his or her work and has mental tricks for handling uncertainty and stress is more prepared to deal with said uncertainty, and over time is more productive and happy. It should also be noted that although these points of discussion are especially relevant for remote software engineering, they can easily be extended beyond this domain of work. These points offer a refreshing idea that success and productivity does not happen to us, but is something that we choose to create.

References

[1] Almost two thirds of women face everyday sexism and racism at work. https://leanin.org/women-in-the-workplace-report-2018/everyday-discrimination-microaggressions. Accessed: 2020-6-3.

[2] Build software better, together.

[3] Definition of productivity. https://www.merriam-webster.com/dictionary/productivity. Accessed: 2020-6-3.

[4] The five myths of Self-Compassion. https://greatergood.berkeley.edu/article/item/the_five_myths_of_self_compassion. Accessed: 2020-6-3.

[5] Gitter. https://gitter.im/. Accessed: 2020-6-3.

[6] The scrum guide.

[7] Myth of developer productivity, Jul 2019.

[8] Torgeir Dingsøyr, Sridhar Nerur, Venugopal Balijepally, and Nils Brede Moe. A decade of agile methodologies: Towards explaining agile software development. *J. Syst. Softw.*, 85(6):1213–1221, June 2012.

[9] William Gant. Remote-First: A guide for organizations - simple programmer. https://simpleprogrammer.com/remote-first-guide/, December 2019. Accessed: 2020-6-3.

[10] Tyler Hakes. How to measure developer productivity, Jun 2019.

[11] Paula Irwin. Knowmail. January 2018.

[12] 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.

[13] Maria Siniaalto and Pekka Abrahamsson. A comparative case study on the impact of Test-Driven development on program design and test coverage. November 2017.

[14] Vanessa Sochat. The sadness of the open source developer. https://vsoch.github.io//2017/sadness-open-source-developer/, December 2017. Accessed: 2020-6-3.