CHAPTER 12

Developers' Diverging Perceptions of Productivity

André N. Meyer, University of Zurich, Switzerland
Gail C. Murphy, University of British Columbia, Canada
Thomas Fritz, University of Zurich, Switzerland
Thomas Zimmermann, Microsoft Research, USA

Quantifying Productivity: Measuring vs. Perceptions

To overcome the ever-growing demand for software, software development organizations strive to enhance the productivity of their developers. But what does productivity mean in the context of software development? A substantial amount of work on developer productivity has been undertaken over the past four decades. The majority of this work considered productivity from a *top-down perspective* (the manager view) in terms of the artifacts and code created per unit of time. Common examples of such productivity measures are the lines of source code modified per hour, the resolution time for modification requests, or function points created per month. These productivity measures focus on a single, output-oriented factor for quantifying productivity and do not take into account developers' individual work roles, practices, and other factors that might affect their productivity, such as work fragmentation, the tools used, or the work/office environment. For example, a lead developer who spends a big part of work supporting co-workers with their inquiries might develop less code in the process

C. Sadowski and T. Zimmermann (eds.), *Rethinking Productivity in Software Engineering*,
https://doi.org/10.1007/978-1-4842-4221-6_12

and would thus be considered less productive when using traditional, top-down measurements compared to developers who focus solely on coding.

Another approach to quantify productivity is *bottom-up*, starting at the productivity of individual software developers to then also learn more about quantifying productivity more broadly. By investigating developers' individual productivity, it is possible to better understand individual work habits and patterns, how they relate to productivity perceptions, and also which factors are most relevant for a developer's productivity.

Studying Software Developers' Productivity Perceptions

There are various ways to investigate productivity from the bottom up. In this chapter, we describe three studies that we conducted using a variety of methods, from very detailed observations to two-week field studies using a monitoring application.

- First, to gather insights into what developers' considered productive and unproductive work, we conducted an online survey with 389 professional software developers, followed by observations and follow-up interviews with 11 developers to corroborate some of the findings of the survey [1].
- To better understand activities developers pursue at work, the fragmentation of their work, and how these activities relate to self-reported productivity, we conducted a two-week field study with 20 professional software developers. For this study, we deployed a monitoring application that logged developers' computer interaction and collected self-reports on their productivity every 90 minutes [2].
- To analyze and compare the situations when developers feel productive, we conducted a further online survey with 413 professional software developers [3].

The remainder of this chapter highlights the most prominent findings. Detailed descriptions of the studies and findings can be found in the corresponding papers.

The Cost of Context Switching

Developers reported that they usually feel most productive when they make progress on tasks and when they have only a few context switches and interruptions. However, observing developers' workdays revealed that they constantly switch contexts, often multiple times an hour. For example, developers switched tasks on average 13 times an hour and spent just about 6 minutes on a task before switching to another one. An example of a task switch is a developer who is switching from implementing a feature to answering e-mails that are unrelated to the previous task. Similarly, when we looked at how much time developers spend on activities–actions they usually pursue at work (e.g., writing code, running tests, or writing an e-mail)–we found out that they usually remain in an activity only between 20 seconds and 2 minutes before switching to another one. This high number of task and activity switches and the high variety of activities and tasks developers pursue each day illustrate the high fragmentation of a developer's work.

Surprisingly, many developers still felt productive despite the high number of context switches. The follow-up interviews with the developers revealed that the cost of context switches varies. The cost or "harm" of a context switch depends on several factors: the duration of the switch, the reason for the switch, and the focus on the current task that is interrupted. A short switch from the IDE to respond to a Slack message is usually less costly than being interrupted from a task by a co-worker and discussing a topic unrelated to the main task for half an hour. Also, short context switches, such as writing a quick e-mail while waiting for a build to complete, do not usually harm productivity, as self-reported by our participants.

Interruptions from co-workers are one of the most often mentioned reasons for costly context switches, especially when they happen at an inopportune moment, such as when a developer is focused on a challenging problem. Chapter 23 presents one possible solution of how developers and other knowledge workers can reduce the number of costly interruptions by visualizing their current focus to the team.

A Productive Workday in a Developer's Life

Investigating how developers organize their time at work and what activities they pursue revealed notable differences. During an average workday of 8.4 hours, developers spend about half of their time, on average 4.3 hours, actively working on their computer. Surprisingly, they spend only about one-fourth of their total work time with coding-related activities and another fourth of their time with collaborative activities such

as meetings, e-mails, and instant messaging. There are also big differences across companies, for example how much time their developers spend reading or writing e-mails. At one of the observed companies, developers spent less than one minute with e-mail each workday, compared to developers at another company where they spent more than an hour.

Relating the activities developers pursue at work with how productive they feel during these activities revealed that productivity is highly individual and differs greatly across developers. The majority of developers reported coding as the most productive activity, as coding allows them to make progress on the tasks that are most important to them. With most other activities, there was no clear consensus about whether an activity is generally productive or not. Meetings were the most controversial activity: more than half of the developers considered meetings as unproductive, especially when they lack goals, have no outcome, or there are too many attendees; the other half of developers considered meetings to be productive. E-mails are considered to be a less productive activity by many developers. However, no single activity is considered exclusively productive or unproductive by all developers. Coding, for instance, was not always considered to be a productive activity, for example when the developer was blocked on a task. This suggests that measures or models that attempt to quantify productivity should take individual differences, such as the context of a developer's workday, into account, and attempt to capture a developer's work more holistically rather than reducing them to a single activity and one outcome measure.

Developers Expect Different Measures for Quantifying Productivity

When we asked developers about how they would like to quantify their productivity, the majority wanted to assess their productivity based on the number of completed tasks but also combine it with other measures. These additional measures include output-related measures, such as the lines of code, number of commits, number of bugs found or fixed, and e-mails sent, but they also include higher-level measures, such as how focused they were during their work, if they were working "in the flow" (or "the zone"), and if they felt they had made any significant progress. Across all measures that developers were asked about, there was no single measure or combination of multiple measures that were consistently rated higher by most developers. This result indicates that there are a variety of aspects that impact the productivity of developers and their feeling of productivity

differently. For example, on days when a developer spends a lot of time working on development task, a measure of the number of work items completed or check-ins made may be appropriate. However, the same measure on days a developer spends most of the time in meetings or helping co-workers would result in a low productivity and high frustration for the developer. Furthermore, the findings suggest that it is difficult to broadly measure productivity without defining specific objectives. We will have to find ways to do measure productivity more holistically, by not only leveraging output measures, but also considering developers' individual abilities, work habits, contributions to the team, and more. Chapters 2 and 3 discuss this further and argue that productivity should be considered not only from the perspective of individuals but also for teams and organizations.

Characterizing Software Developers by Perceptions of Productivity

The differences in how developers feel about productivity makes it also more challenging to determine meaningful actions that could help increase productivity on a team or organizational level. One way to better understand differences and commonalities in developers' perceptions of productivity is to investigate if we can find patterns or group developers with similar perceptions. Analyzing productivity ratings from hourly self-reports during three workweeks, we found that developers can roughly be categorized into three groups that are similar to the circadian rhythm: morning person, afternoon person, and low-at-lunch person, as visualized in Figure 12-1. The curved regression line in the three figures shows the overall pattern of what part of the day an individual developer typically felt more or less productive with the shaded area showing the confidence range. Morning people were rare in our sample, with only 20 percent of all participants. The biggest group were afternoon people (40 percent), who may be those who are industrious later in the day or who feel more productive as a result of having the majority of their workday behind them. These results suggest that while developers have diverse perceived productivity patterns, individuals do appear to follow their own habitual patterns each day.

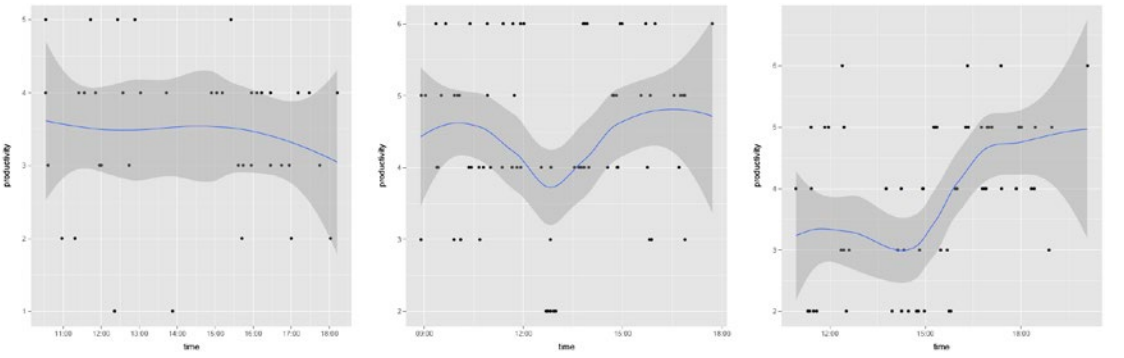

Figure 12-1. *Three types of developers and their perceptions of productivity over the course of a workday*

In another effort to group developers with similar perceptions of productivity together, we asked participants to describe productive and unproductive workdays, rate their agreement with a list of factors that might affect productivity, and rate the interestingness of a list of productivity measures at work. We found that developers can be clustered into six groups: social, lone, focused, balanced, leading, and goal-oriented.

- The *social developers* feel productive when helping co-workers, collaborating, and doing code reviews. To get things done, they come early to work or work late and try to focus on a single task.
- The *lone developers* avoid disruptions such as noise, e-mail, meetings, and code reviews. They feel most productive when they have little to no social interactions and when they can work on solving problems, fixing bugs, or coding features in quiet and without interruptions. To reflect about work, they are mostly interested in knowing the frequency and duration of interruptions they encountered. Note that this group of developers is almost the opposite of the first group (the social developer) in how productive they feel when encountering social interactions.
- The *focused developers* feel most productive when they are working efficiently and concentrated on a single task at a time. They feel unproductive when they are wasting time and spend too much time on a task because they are stuck or working slowly. They are interested in knowing the number of interruptions and length of focused time.

- The *balanced developers* are less affected by disruptions. They feel unproductive when tasks are unclear or irrelevant, when they are unfamiliar with a task, or when tasks are causing overhead.
- The *leading developers* are more comfortable with meetings and e-mails and feel less productive with coding activities than other developers. They feel more productive when they can write and design things, such as specifications. They do not like broken builds and blocking tasks, preventing them (or the team) from doing productive work.
- The *goal-oriented developers* feel productive when they complete or make progress on tasks. They feel less productive when they multitask, are goal-less, or are stuck. They are more open to meetings and e-mails compared to the other groups if they help them achieve their goals. In contrast to focused developers, goal-oriented developers care more about actually getting stuff done (i.e., crossing items off the task-list), while focused developers care more about working efficiently.

Each developer can belong to one or more of these groups. The six groups and their characteristics highlight differences in developers' productivity perceptions and show that their ideal workdays, tasks, and work environments often look differently. We can further use these findings to tailor process improvements and tools to the different types of developers, as discussed in the next section.

Opportunities for Improving Developer Productivity

Developers and development teams might benefit from these findings in various ways. On the individual level, we could build self-monitoring tools that allow developers to increase their awareness about productive and unproductive behaviors and use the insights they gain to set well-founded goals for self-improvements at work (see Chapter 22).

These approaches should provide a variety of measures and support developers in getting insights into individual aspects of their work, such as identifying productive or unproductive work habits or identifying external or internal factors that have the biggest impact on their productivity. In addition to self-monitoring that has been shown to motivate positive behavior changes in other fields (e.g., physical activity and health), supporting developers with setting goals to improve themselves at work through actionable insights might be a next step toward fostering productivity. Maybe one day, we can further build virtual assistants, such as Alexa for Developers, that recommend (or automatically take) actions, depending on the goals of developers or based on the productivity patterns/roles/clusters of developers. For example, such a virtual assistant could block out notifications from e-mail, Slack, and Skype during coding sessions to avoid disruptions for the "lone developer" but allow them for the "social developer." Or they could recommend the "focused developer" to come to work early to have a few hours of uninterrupted work time or suggest the "balanced developer" to take a break to avoid boredom and tiredness.

By knowing the trends of developers' perceived productivity and the activities they consider as particularly productive/unproductive, it might be possible to schedule the tasks and activities developers must perform in a way that best fits their work patterns. For example, if a developer is a morning person and considers coding particularly productive and meetings as impeding productivity, blocking calendar time in the morning for coding tasks and automatically assigning afternoon hours for meeting requests may allow the developer to best employ their capabilities over the whole day. Or, it could remind developers to reserve slots for unplanned work or interruptions at times where they usually happen.

Our studies also revealed that interruptions, one specific type of a context switch, are one of the biggest impediments to productive work. Productivity could potentially be improved on the team level by enhancing the coordination and communication between co-workers, depending on their preferences, availabilities, and current focus. For example, on the team level, quiet, less interruption-prone offices could be provided to the "lone developers" and "focused developers," and "social developers" who feel more comfortable with discussions every now and then could be seated in open space offices. Alternatively, interruptions at inopportune moments could be reduced by visualizing the developer's current focus and concentration to other developers using an external cue. Hence, at times when the developer is "in the flow" or is usually most productive, expensive interruptions could be postponed to a more opportune moment (see Chapter 23).

Key Ideas

The following are the key ideas from this chapter:

- Different software developers experience productivity differently, which is why they do not agree on how to measure productivity.
- Most developers follow their own habitual patterns each day and are most productive either in the morning, during the day (and not at lunch), or in the afternoon.
- Measuring developer productivity should not only include output measures but also include measures inherent to developers' abilities, workdays, work environments, and more.

References

[1] André N Meyer, Thomas Fritz, Gail C Murphy, and Thomas Zimmermann. 2014. Software Developers' Perceptions of Productivity. In Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 19–29.

[2] André N Meyer, Laura E Barton, Gail C Murphy, Thomas Zimmermann, and Thomas Fritz. 2017. The Work Life of Developers: Activities, Switches and Perceived Productivity. Transactions of Software Engineering (2017), 1–15.

[3] André N Meyer, Thomas Zimmermann, and Thomas Fritz. 2017. Characterizing Software Developers by Perceptions of Productivity. In Empirical Software Engineering and Measurement (ESEM), 2017 International Symposium on.