jasoncrawford.org /10x-engineers

"10x engineers": Stereotypes and research

8-10 minutes

July 14, 2019 · 5 min read

I was going to write about something else today, but it looks like everyone on Twitter is talking about "10x engineers" again.

Do 10x engineers exist? And what does the term even mean, anyway?

The topic sparks strong emotional reactions in people, because it touches deep ideological topics: whether some people are more talented than others, why that is (and whether it's inherent or changeable), and how we should treat people if so. Thus the acrimonous fights on social media: On the one side, people say that 10x is a myth, that it's all based on stereotypes, and that anyway aren't there more important things like whether you can write documentation or mentor other engineers or just be a nice person? On the other side, people roll their eyes and say *of course* there are 10x engineers and anyone who doesn't acknowledge this obvious fact has been blinded to reality by social justice propaganda or is a loser who doesn't want to admit their own inferiority.

Let's try to get real. And to start, let's get empirical.

What to know about the research

The best explanation of the concept and its basis the article "Productivity Variations Among Developers and Teams: The Origin of 10x" by Steve McConnell. Here are the key things to know about "10x engineers":

- **10x refers to the difference between the best and worst developers, *not the best and average*.** This makes a big difference. To my mind it's easier to believe that the best developers are ~3x more productive than the average, and the worst are ~3x less productive. That could get you to a 10x overall spread. (Note that therefore it doesn't make any sense to talk about a "0.5x engineer", because the worst developers by definition are 1x.)

  Maybe we should just redefine "x" to be the average, call the best ones "3x engineers", and end this whole debate?

- **The 10x concept is based in research, but the research is not perfect.** McConnell in the link above describes the studies that have been done (if you want to go deep, see also this critique and McConnell's rebuttal). There is reason to be critical of the studies and their relevance to today: The sample sizes are relatively small, and they aren't always well-controlled. Some of them were done decades ago (the first was 1968), when computers, programming languages, and development tasks were significantly different. They sometimes use dubious measures of productivity, such as lines of code per day, but they also use good measures like time to completion, and in some contexts lines of code is properly considered a negative (for a given task, fewer lines is considered to be better).

  Overall, I think there is significant evidence of wide variations in productivity among individuals and (a bit less so) among teams.

- **10x is a rough average.** Different studies and measures find different ranges, generally between 5x and 25x. This combined with the limitations of the studies just discussed means we can't say anything more than "roughly order of magnitude". "10x" is not precise; it's just a compact way of remembering that differences in productivity exist and are large.

- **The studies only compare differences among developers who *actually complete the task.*** Their figures don't take into account the people (~10% in some studies) who didn't even finish. Nor can they take into account the real-world cost of software that is nominally completed but is so buggy, flaky, or hard to maintain that it has to be rewritten by someone else. And that's not

even mentioning the cost of bugs that affect sales and lose money for a company, or even put lives at risk. All these factors, though, would just reinforce the basic point.

- **The 10x figure is narrowly about specific measures of engineering productivity.** It is not intended to fully capture an engineer's value to an organization and can't be used for that purpose. That said, these are still meaningful and important measures.
- **The studies don't address the causes of the differences,** to my knowledge. McConnell's survey, at least, doesn't address important questions such as: Is an individual's productivity level stable over time? Does it vary by environment, and how much does the work environment affect productivity? Does it vary randomly from assignment to assignment, by the luck of a particular moment of insight? Does it grow over time? Can it be learned or taught? (He says that the initial 1968 study "found no relationship between a programmer's amount of experience and code quality or productivity", but that doesn't mean that productivity *can't* grow over time, just that it doesn't *necessarily* do so.)

What I believe

Here's what I think based not only on the research but on my own anecdotal experience and personal worldview:

- **Differences in productivity are real, large, and important, and probably underappreciated.** If the difference is not quite "10x", then it's large enough to matter a lot. I believe this in part because of the research but in part because the phenomenon is much wider than software. McConnell himself points this out, citing a study by Norm Augustine that "found that in a variety of professions—writing, football, invention, police work, and other occupations—the top 20 percent of the people produced about 50 percent of the output, whether the output is touchdowns, patents, solved cases, or software."
- **Work environment matters a lot.** In a real-world setting, the kind of productivity that matters is strongly conditioned by the *context* of the work. Are engineers given clear goals and priorities? Do they buy in? Are they motivated? Do they trust each other, and their management? Are they allowed to focus? Are they randomized with meetings? Or with fighting production fires? Do they have good infrastructure and tooling? Etc.
- **Productivity is a combination of inherent traits and acquired skills.** That is to say, it can be learned, partially. The acquired skills include a range of things from specific debugging tools to general patterns of thinking and problem-solving. The inherent traits have a lot to do with intelligence and other thinking patterns that we can't (or don't yet know how to) identify and teach.
- **Productivity is not strongly correlated with experience.** There are *very* productive junior engineers and very mediocre (or worse) senior ones. So even if productivity can be learned, that learning doesn't happen automatically, or even happen very often in our industry. Productivity skills are rarely taught, and the kind of people who learn them independently are also the kind of people who were good to begin with.

So what?

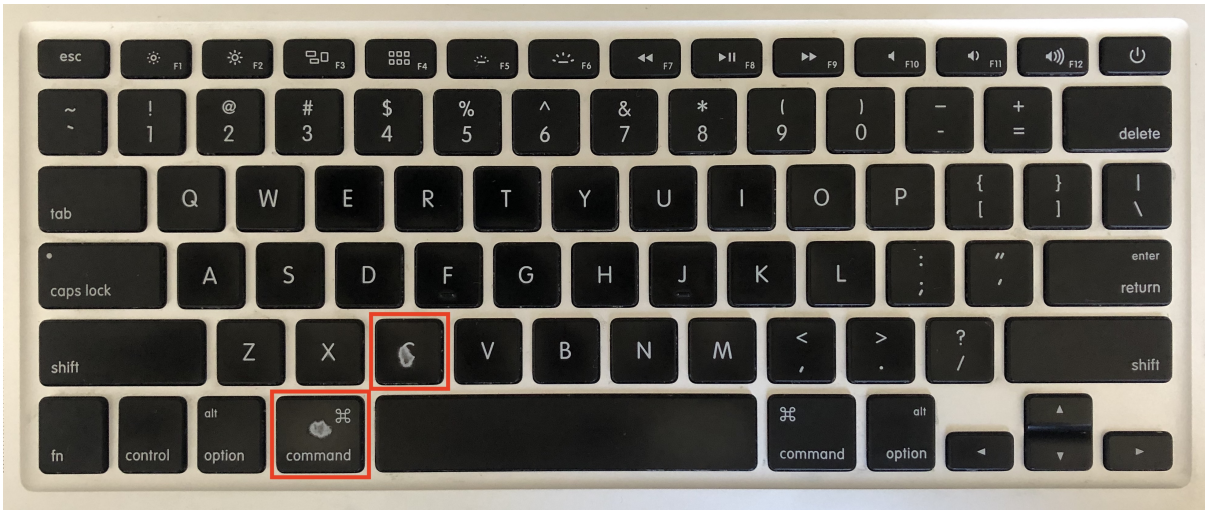
What is the 10x debate actually about? The emotional impact of the question is its implication for how we hire, how we reward people, and how we treat individuals.

Here are my conclusions:

- **Hiring matters.** Work hard to attract, discover and close the best talent.
- **Environment matters.** Work hard to create a great work environment, and if you're seeing general problems with productivity, it might be an environment problem.
- **Reward productivity.** People who create more, have earned more.
- **BUT, don't assume productivity is inherent in an individual.** Someone who was unproductive in one setting might be productive on a different project or in a different team or company. Diagnose the problem before you dismiss the individual.

- **Train engineers in productivity skills.** I don’t yet know how effective this can be, but I think we as an industry are not even trying here (and I include myself in this).
- **Productivity doesn’t excuse bad behavior.** This one ought to go without saying.

And there are no special tricks to “spot” a 10x engineer (despite the bizarre, ridiculous Twitter thread that sparked the current chatter). They aren’t fairies or leprechauns, and you can’t identify them by the color of their terminal, the wear on their keyboard, or any other arbitrary stereotypes.



My keyboard, after 6 years of use. I guess the only thing I do a lot is copy. I promise it’s not ALWAYS from Stack Overflow.