hackernoon.com

How to Become a 10x Dev: An Essential Guide

TrueSeniorDev

19–24 minutes

> - Software development is a team sport. Your individual performance doesn't matter as much as the performance of your whole team and company.
> - By improving how you work, you only *ADD* to your team's performance. By improving how everyone works, you *MULTIPLY* your team's performance.
> - Multiplying the whole team's performance will put you on a much faster lane for being noticed and promoted than being a crazy efficient individual contributor.
> - Boosting other people's performance requires a different mindset and approach than boosting your individual performance. In this article, I'll show you 16 proven ways to multiply the performance of your team (or even the whole company).

The Common Misconception of 10x Dev

If you google the term "10x developer", you'll get a lot of results related to individual performance.

Check out the quotes from a few of the topmost results that I've got:

- *"A ten times developer is the developer who is ten times more efficient than the average developer on a team."*
- *"These are the people you want to solve your problems; they'll do it in 1/10th of the time, with 1/10th of the number of lines of code."*
- *"I am accustomed to being on teams where I am doing 60% of the work with 7-8 developers on the team."*

Statements like these make me cringe.

This. Is. Not. How. We. Build. Modern. Software.

Software development is a team game. Or even a team of teams game. And the game is scored by business outcomes, not raw outputs. By being *effective* as a team and organization, not *efficient* as an individual.

As a VP of engineering, I'm not looking for competitive coders. I don't care how many lines of code you crank in what time. And a single developer doing 60% of the team's work is not something I'd brag about but a dysfunction I'd consider my priority to fix.

So, who I *am* looking for, then? Let's consider the following math:

The Simple Math of Working In a Team

If you improve only your own output, you *ADD* to the output of your team.

But if you improve every team member's output, you *MULTIPLY* the output of your team.

And multiplication beats addition pretty quickly.

Even if you truly are a mythical 10x dev, you'll only increase your team's output by 10 "units", no matter the size of the team. But if you improve the whole team's output by just 2x, you'll increase it by 10 "units" for a team of only 10 people. And by 100 "units" for a team of 100 people.

And we're talking about an extreme situation when you are 10 times more efficient than *EVERY* other developer on the team. In reality, multiplication will beat addition for teams much smaller than 10 people.

Plus, we're talking only about the raw output: how hard do you push, but not in which direction you push. And the direction *the whole team pushes in* is critical.

Your Team's and Your Individual Performance Are Connected

It doesn't matter if a motorboat has a 100 or 1000-horsepower engine if this engine pushes the boat sideways or backward, not forward. If you push in the wrong direction, you may not only waste your own output - you may undermine the work of your whole team.

You can become a *-10x* developer.

And this relationship is two-sided.

Even if *you* push in the right direction, your god-mode 10x productivity can be negated if the rest of your team is pushing in the opposite direction.

You need to help them, so they can help you. Cancel the weaknesses of your team so they don't cancel your performance.

It may sound counter-intuitive, but focusing on making your whole team more effective is often a better way to boost your individual performance rather than focusing only on your own output.

That's why I and other managers aren't looking for individual efficiency as much as for the capability to positively impact your team. And the broader impact you make - the more people you help go fast in the right direction - the more value you generate for your company.

But don't just take my word for it.

Let's take a look at engineering career ladders at a couple of well-known companies:

CircleCI

CircleCI uses a 6-level ladder (E1-E6): Associate Engineer, Engineer, Senior Engineer, Staff Engineer, Senior Staff Engineer, and Principal Engineer.

Levels E1-E3 focus on the execution of work. E1 within task, E2 within epic/project, E3 within team.

Levels E4-E6 utilize skills to scale and generate leverage. They facilitate, guide, and mentor others. E4 within team and with team's business stakeholders, E5 across several teams, and E6 across organization.

SOURCE: https://docs.google.com/spreadsheets/d/131XZCEb8LoXqy79WWrhCX4sBnGhCM1nAIz4feFZJsEo/edit#gid=0

Carta

Carta uses a 7-level ladder (L2-L8). In their own words:

> It’s easy to articulate the single most important thing for leveling: your impact on the company. We can sum up the entire system by describing the (rough) impact we expect employees to have as they progress: on tasks (L2), on features (L3), on problems (L4), on teams (L5), on the organization (L6), on the company (L7), and on the industry (L8).

SOURCE: https://medium.com/building-carta/engineering-levels-at-carta-d33db2a55a20

Spotify

Spotify doesn't care that much about external titles like senior, staff, or principal developer. They are very flexible about them and let employees choose what makes the most sense for them. But internally, they use a 4-level ladder organized by what they call "scopes of impact".

And they describe these 4 levels ("steps") like this:

> We have identified four Steps in your career path at Spotify. Each Step is marked not only by increased responsibility, but also by your increased impact within tech: Individual Step, Squad/Chapter Step, Tribe/Guild Step, Technology/Company Step.

SOURCE: https://engineering.atspotify.com/2016/02/spotify-technology-career-steps/

Dropbox

Dropbox uses a 7-level ladder (IC1-IC7): Software Engineer 1-4, Staff Software Engineer, Principal Software Engineer, and Senior

Principal Software Engineer.

This is how they describe the "extent of influence" for each level:

- IC1: I work within the scope of my team with *specific* guidance from my manager.
- IC2: I work primarily within the scope of my team with *high-level* guidance from my manager.
- IC3: I work primarily with my direct team and cross-functional partners while driving cross-team collaboration for my project.
- IC4: I am a strong leader for my team with my impact beginning to extend outside my team.
- IC5: I am increasingly influencing the roadmaps of other Dropbox teams to achieve business-impacting goals.
- IC6: I typically influence the technical strategy of a group.
- IC7: I typically influence the department and company-wide strategy to achieve business-impacting goals.

SOURCE: https://dropbox.github.io/dbx-career-framework/

I have chosen these 4 companies as they describe their ladders most succinctly, and thus, can be quoted almost directly. But a similar pattern repeats widely across the whole industry.

So what gets you promoted in all these companies? What do they value the most?

Again, the area of the impact that you make.

The bigger chunk of an organization (and the more people) you positively influence, the more valuable you'll be for the company - and the more recognition you'll get.

Operating at such a level may sound scary. But boosting your whole team's performance by 2 or 3x is often less daunting than it sounds. And easier than dialing up your individual performance to 10x.

Ok, so how can you do it?

How to Impact the Performance of the Whole team or Company

It's hard to be completely exhaustive, as there are *many* ways you can positively impact other people's performance. But let's explore several of them so you can get a good gist and build the mindset that will let you come up with further ideas on your own.

Start small. Discuss stuff with your teammates. Spread your knowledge through code reviews, pair programming, and team meetings. Lead by example, through your work and behavior.

Spread your knowledge not only about coding but about anything that may impact your team's performance: processes, communication, and time management.

Extend your impact to more people through brown bags, workshops, and blogging. Take opportunities to collaborate with other teams.

Finally, when you're widely recognized as an expert, take responsibility for managing company-wide learning programs or leading communities of practice.

2. Bring New Knowledge Into the Company

Don't be a one-time sensation. Don't rely only on your previous experience.

Learn continuously. Talk to colleagues outside your company. Read. Go to conferences. Stay up to date with the industry's state-of-the-art. Research what other companies are doing. Experiment with new techniques and libraries.

Be the source of innovative knowledge that will help your team reach the next level.

3. Plan and Coordinate Projects

No matter what process you use, software development is, in the end, a stream of big and small projects. How well they are analyzed, broken down, planned, and executed has a tremendous impact on the performance of the teams running them.

Volunteering to prepare and lead projects is a great way to impact multiple people's performance.

Start by taking responsibility for one part of the process: requirements analysis, implementation plan, or task breakdown.

Dial it up by owning the process end-to-end, overseeing the whole project execution.

And for maximum impact, take responsibility for coordinating big, cross-team projects.

4. Take Ownership of a Part of the Codebase or Product

In complex software systems, there are many "moving parts" that need to be maintained over a long time: modules, libraries, subsystems, products, services, APIs, tools, documentation, pipelines, and so on. It's not easy to maintain them well without clear ownership.

And how well they are maintained has a huge impact on the performance of everyone using them - which, for some cornerstone modules, may even mean everyone in the whole company.

By taking ownership of a part of your company's codebase, product, or subsystem and keeping it in great shape, you can make an impact that will be hard to miss.

On a smaller scale, you can maintain something internal to your team, for example, a small code module or web service.

On a bigger scale, you can maintain something fundamental to the whole product and company, like a central design system or customer-facing API.

5. Improve Tooling

Efficient tools make a tremendous difference in how fast a team can go. And there are so many things in our work that can be optimized, automated, or used in a more skillful way.

Solid CI/CD pipeline. Code formatting and linting. Fast test suite. Automated code and test data generation. More efficient code navigation. Utilizing the full power of your IDE. Better local, test, and staging environments. Well-configured project management system.

Robust tooling for debugging, monitoring, and logging. Better discoverability of components and shared libraries. Automating and connecting your workflows (Github, Slack, Trello, etc.). The list could go on and on.

By improving the tooling, automating repetitive tasks, or even just spreading the knowledge of how to use existing tools better, you can greatly impact the performance of your team and the whole company.

6. Improve Codebase and Architecture

Put yourself in the shoes of colleagues who will visit the project after you. Easy-to-understand code and architecture can make or break the whole team's performance. Make them cleaner, simpler, easier to navigate and debug, and less error-prone.

Even relatively small improvements, to one component or module, can considerably boost the performance of your team. And the more global improvements - for example, to product-wide code conventions - can impact even the whole company.

7. Help Your Team Go in the Right Direction

Rallying people together to push towards a single, right direction is one of the most powerful ways to multiply the whole team's performance. And you can influence it much more than you think.

First of all, understand the right direction yourself. Put effort into understanding your company's goals, business, and customers. This will allow you to understand what problems need to be solved, and come up with better solutions to these problems.

Second, help your team go in the right direction. Share and document what you learned. Help your team track relevant metrics, gather feedback, and work in a more iterative, agile way. Become a trusted consultant for your product manager.

Third, help coordinate with the other teams and stakeholders, so the whole company goes in a single direction.

8. Support Non-Developers

Software isn't built only by software developers. It's a concerted effort of developers, designers, testers, product managers, analysts, data scientists, user researchers, and various business stakeholders (customer support, marketing, sales, finance).

Support them. Work closely together. Put effort into making their lives easier, into helping them go fast, and into coordinating your efforts so the whole cross-functional group works smoothly together.

This will allow you to impact your organization at an even broader scope than the engineering team alone.

9. Improve Methodologies and Processes

How your team and organization work has a fundamental impact on performance. And it's not reserved only for managers and scrum masters. As a developer, you can, too, influence company processes a lot.

First, educate yourself. Understand the principles of empiricism, iterative development, product discovery, and Agile. Get to know modern technical approaches like CI/CD or feature flag-based development. Learn how modern cross-functional teams function.

Second, observe with a critical eye, proactively look for opportunities for improvement, and take the initiative in shaping the process. Be active inside and outside of your team. Participate in process-related discussions. Champion implementing new ideas.

You'll be able to multiply the performance of many people and teams.

10. Lead Technical Initiatives

Many technical improvements require a concerted effort of the whole team over a longer time: successively replacing the old framework with a newer one, migrating to a new set of coding conventions, and gradually refactoring a critical part of the codebase.

If they aren't consistently managed and pushed forward, such initiatives usually quickly fizzle out. And their outcomes are mediocre at best.

Volunteer to lead such initiatives. If you plan and manage them well, if you make it easier for other people to contribute, and if you coordinate the efforts around them and see them through to completion, you can make a substantial impact on the future performance of your team and organization.

11. Improve Communication and Transparency

Clear, transparent communication is essential for effective teamwork. It determines how well people coordinate, how much they trust each other, how good decisions they make, and how well they understand their objectives.

And this applies at all levels: inside a team, between teams, between different roles and departments, and between employees and management. The quality of communication can hinder or unlock the whole organization's performance.

And you can do a lot to improve it.

Share your status with the other teams. Maintain your team's and company's documentation, roadmaps, and wikis in a good state. Help your team stay informed by proactively pulling information from the other teams.

Ask management to clarify your team's goals and update them on your progress. Propose improvements to company-wide communication standards and channels (Slack, etc.). Initiate cross-team coordination meetings whenever necessary.

12. Exemplify and Promote the Culture

There's no single right culture. Different companies conduct themselves differently with a similar level of success. But if different people, teams, and departments in the same company culturally clash, it kills productivity.

Understand your company's culture. Exemplify and promote it. Be conscious of your behavior and communication patterns.

It impacts your company's performance more than you think.

13. Motivate Others

Sometimes, reaching higher performance is just a matter of motivation. The willingness to push harder. Feeling more hungry for success. Having the right attitude.

You don't have to be a manager to influence your team's morale. Lead by example. Spread good vibes. Show enthusiasm, optimism, and grit. Make the work fun. Stay calm and composed in a crisis. Rally your team to push a little bit more every day.

Such behaviors are viral and frequently work better than attempts to "empower" people from the top. And they easily spread outside of your team, which will let you make a broad impact.

14. Help With Recruitment and Onboarding

Another way to boost an organization's performance is through recruitment.

If you can, get involved in the recruitment process. Participate in the interviews. Help prepare recruitment challenges. Review CVs.

And even if you can't participate in the recruitment process itself, there's still a lot you can do to help bring new talent on board. Refer your colleagues. Promote your company through blogging, discussion forums, speaking at conferences, and networking. Leave a positive review on Glassdoor.

Take responsibility for onboarding and mentoring new hires.

15. Solve Complex Problems (and Then Spread the Knowledge)

If you can solve a problem no one else is able to solve, such that will give your company a competitive advantage, you can generate a ton of value.

This may sound like a task for a lone, 10x genius, but it is still about the team. If you won't spread the knowledge about your brilliant solution, if you won't design it so it's easy for others to reuse, you'll become a bottleneck and a potential single point of failure instead of a multiplier.

You'll put your company at risk, not bring value.

But if you can solve complex problems in such a way that everyone understands your solution and is able to build on top of it, you can become a game changer.

16. Provide Technical Direction and Advice

Your company often needs to cope with decisions with a huge, long-lasting impact. Choosing a technology stack. Choosing a cloud vendor. Making build vs buy decisions.

Even more often, smaller questions pop up: What would be the rough size of the project? Is it technically feasible? What are the possible solutions to this problem?

Become a go-to advisor for your team and the whole business. Learn the industry's landscape in-depth. Stay up-to-date with the newest developments and trends. Get good at research. It'll let you bring a lot of value to your company.

"But Can I Really Do All of This as a Developer?"

Can you really own subsystems, lead projects, or make build vs buy decisions? Isn't it the responsibility of architects, managers, and tech leads?

Yes, you can! In any well-run company, your manager or tech lead will be glad to delegate as much as possible to you. This is actually their job. It helps you grow and your team perform better, which are the most important responsibilities of a manager.

Some of the higher-impact stuff may indeed require more experience - nobody will let a junior straight out of a coding bootcamp own a mission-critical subsystem used by all the teams in the company. But all the impacts we discussed have a gradual progression.

Start with smaller initiatives in your team, and successively expand the scope of your impact. Work with your lead or manager to find more growth opportunities. Be on the constant lookout for such opportunities yourself, in your team, and across the company. Get noticed and build a reputation.

This will set you on a fast track to making a bigger and bigger splash. To become not only 10x but even a 100x developer. And - if this is your ambition - to eventually become a tech lead yourself.

Bottom line

Yes, you need to be a solid developer. It's hard to be a multiplier for others while doing crappy work yourself. But don't sweat it if you are really 10x, or 5x, or just a solid 1x. Shift your mindset from personal efficiency to the effectiveness of the whole team and company.

And when you do, only the sky (and your company's size) is the limit. You can become not only 10x dev but even 100x and more.