Delegation as Art

**BE SOMEONE WHO
MAKES EVERYONE
ELSE BETTER**

KATE MATSUDAIRA

When I started my career as a junior engineer, I couldn't wait to be senior. I would regularly review our promotion guidelines and assess my progress and contributions against them. Of course, at the time I didn't really understand what being senior meant.

Being a senior engineer means having strong technical skills, the ability to communicate well and navigate ambiguous situations, and most important of all, the ability to grow and lead other people.

Leadership isn't just for managers anymore.

I have worked with many genuinely talented engineers throughout my career. I have noticed that truly great engineers bring forth the capabilities of everyone around them. Having them on the team makes everyone brighter, more productive, and more motivated.

The difference between being an engineer and being a senior engineer often has to do with the ability to mentor and grow your teammates. Mentoring is all about *teaching* and *delegation*.

Adding senior to your title means that you have the most experience (whether that is industry experience working with a specific set of technologies or methodologies, or simply legacy knowledge of the systems you are using). And because you are senior, part of your job is imparting that knowledge to others.

When done well, it is an amazing thing:

- ➔ All of a sudden you have time to work on new project XYZ instead of being burdened with legacy operations, support, and fixes.
- ➔ You get to watch your colleagues grow and learn, taking on more responsibility and building their skills.
- ➔ Everyone is more productive and hard-working because they are able to learn from past mistakes and take on new challenges.
- ➔ Wool!

Taking on the role of teacher can also be very difficult:

- ➔ When you were the only one who knew system ABC, you were so important. Everyone recognized you and came to you for help—and it felt good to be needed.
- ➔ It can be fun to work on stuff that you are the expert on—it is nice to be able to tackle a task you know how to do (blindfolded or with one hand tied behind your back).
- ➔ “If I hand this over to someone else, I am worried they are going to screw up all of the great work I have done.”
- ➔ When it comes to delegating, it seems like it can take much longer to explain how to do something to someone else than it would take you to get the work done yourself—and who doesn't love to be efficient?

You have to focus on growth and education, not necessarily efficiency.

Even though you know it is beneficial, delegating is hard. You not only have to be able to trust others to get the work done, but you also have to know how much information to give them that will enable them to make progress. This can be really hard to do on gnarly technical problems. How do you delegate tasks and teach others about something that seems very un-delegatable?

I have seen this many times, and from all of my experience I've come up with some suggestions.

1. Change Your Mindset. You have heard that old saying, "Give a man a fish and you feed him for a day; teach a man to fish and you feed him for a lifetime." Well, it applies in your career, too. If you think that taking the time to explain something isn't worth it, then you will never be able to teach anyone else. Sure, it takes more time than doing the task yourself, but that time is valuable to the person you're teaching. You have to focus on growth and education, not necessarily efficiency.

2. Start small. This is the key to successful mentoring and delegating. You want to start with something that is easy to do—for example, fixing a small bug, writing test cases, or adding documentation. When you start with something very tractable, it gives the other person a chance to get his or her feet wet and a moment to peek inside the inner workings without getting overwhelmed. When you want to train someone on something new, try to find small things that aren't urgent—iterative improvements that would be an easy introduction into your realm of responsibility; repeatable things.

3. Lead them to figure it out. It's easy to give people answers when they ask you questions. Next time this happens, resist the urge to tell them how to solve the problem. Instead ask them questions like:

- ➔ What have you tried so far?
- ➔ What have you considered trying?
- ➔ What did you research? Did you find anything?
- ➔ Have you tried looking up _____? (with the blank representing what you would look for to find the answer).

You want to show them the steps you would take and lead them through your thought process—how you would solve the problem—instead of simply providing the solution.

4. Help them come up with a plan. As you give people more responsibility (for example, building a new feature), don't just let them go out on their own. Instead, have them draft a plan and bring it back to you. Go over their approach together. Give them feedback (ideally, by asking good questions instead of just giving them answers). When you draft the idea together, you will feel more confident in their approach and they will be able to leverage your expertise and direction.

5. Set up some guardrails. Whenever someone takes on new responsibility in your team or organization it is good practice to set up some checks to make sure that things go smoothly (especially if you are handing over a mission-critical part of software or operations). Figure out what information you need, and how frequently, so that you can feel confident things are where they should be. For example, you could set up daily status meetings to go over progress, biweekly code reviews to go over implementation, or weekly one-on-

People learning something new are unlikely to get it right the first time.

ones with other people on the project. Try to identify what information you need to feel good about the progress and then the best way to get that information without being overbearing or micromanaging the details.

6. Don't look for perfection. People learning something new are unlikely to get it right the first time. And when it comes to code, it seems like programmers are more like artists, with each individual interpreting a task a little bit differently. Make sure you are open to these differences and accept that other people may not do things the same way you would. If you are really worried about quality, specify your definition of quality up front when you assign the task.

7. Create a team of leaders. Last year I read the book *Turn the Ship Around!: A True Story of Turning Followers into Leaders* by L. David Marquet (Portfolio, 2013) and loved it. One of the most substantial lessons I learned from the book was to adopt a leader:leader model instead of a leader:follower model. Being a good leader means allowing the people around you to be the experts in their domains. My favorite concept from Marquet's book, which truly changed the way I lead, was this: when someone on my team asks me a question, instead of giving them the answer, I ask them, "What do you think I would say?" Most of the time their answer aligns with my thinking, but when it doesn't, I realize that they are missing some important information or context I have neglected to share. This transformed my team; they stopped asking me questions and instead started making recommendations and asking for permission. How can you encourage your teammates to take more ownership and lead their own arenas?

Learning to coach and delegate is not easy, but once you build these skills, you will become even more valuable to your team. You won't just be the expert; you will be someone who makes everyone better—and isn't that the type of person you would want to work with?

LOVE IT, HATE IT? LET US KNOW feedback@queue.acm.org

Kate Matsudaira is an experienced technology leader. She worked in big companies such as Microsoft and Amazon and three successful startups (Decide acquired by eBay, Moz, and Delve Networks acquired by Limelight) before starting her own company, Popforms (<https://popforms.com/>), which was acquired by Safari Books. Having spent her early career as a software engineer, she is deeply technical and has done leading work on distributed systems, cloud computing, and mobile. She has shown herself to be more than just a technology leader, however, by managing entire product teams, research scientists, and building her own profitable business. She is a published author, keynote speaker, and has been honored with awards like Seattle's Top 40 under 40. She sits on the board of acmqueue and maintains a personal blog at katemats.com.

Copyright © 2016 held by owner/author. Publication rights licensed to ACM.