Managing Programmers, with Ron Lichty

Nate Black

From the Editor
In episode 306 of Software Engineering Radio, veteran software manager Ron Lichty joins Nate Black to share his insights on managing software engineers. The title of Ron’s book—Managing the Unmanageable: Rules, Tools, and Insights for Managing Software People and Teams—suggests the challenges inherent in this calling. Nate and Ron delve into what about it is hard, how to grow as a manager, and what makes highly performing teams. They visit themes including understanding the people in your group, building a team out of individuals, giving feedback, empathy, and developing extroversion skills for introverts. The portions omitted here for space reasons cover Ron’s career, how to transition from the technical to the management career path, and what we can learn from the Chicago Bulls championship basketball teams. To hear the full interview, visit se-radio.net or access our archives via RSS at feeds.feedburner.com/se-radio. —Robert Blumen

Nate Black: Ron, your book Managing the Unmanageable talks about programming managers in contrast to other kinds of managers. What’s a programming manager?

Ron Lichty: A programming manager is a manager who manages programmers. My coauthor Mickey Mantle and I think that challenge [comes about because] programmers are a unique bunch. We refer to ourselves as software engineers, and, while there are engineering aspects to what we do, there’s also an art to it.

There is a huge crossover between software developers and musicians. Music has both a left-brain and a right-brain kind of aspect to it. It has a very analytical way that it’s put together and a very artful way to deliver it to delight customers. I have friends who are novelists who say, “Writing software is a lot like writing a novel.” You start with a blank page and craft it from there.

When we’re in the software business, our goal is to delight customers. But we have to be analytical about it as well as artful. That combination of engineering and and craft is unusual.

Part of the challenge of being a programming manager is the nature of the work itself. But part of it is that programmers are different. Could you talk more about that?

The challenge is that not only are we different from everyone else, we’re also different from each other. Programmers, being both engineers and craftspeople, are both left-brained and right-brained. [Managers] have to motivate and enable both parts of brains because those are both parts of who programmers are.

We also have to think about how programmers differ from each other, not just how they differ from other groups of people. There are morning people and night people. There are farmers and cowboys. Every
programmer is unique. The challenge for managers is to figure out the uniqueness of each person who works for us and then to truly support and enable each of those qualities in each person.

Is it correct to say that the manager is responsible only for managing people and their careers, or are there other responsibilities? For instance, do managers write code?

There are plenty of managers who do write code. However, if you’re managing people and you’re writing code, you then are likely to do at most one of those things well—possibly neither. They’re very different qualities.

One of the rules of [career path transition] is that the thing that makes you successful at one level gets in your way at the next level. That’s exactly the opposite of what you need to do as a manager. As a manager, you need to not only put a welcome mat out but also proactively seek out interruptions from the people who work for you: your peers, your boss. It’s really a people kind of job with a technical background.

Why is programming management hard?

To be a good programming manager, you almost have to have been a programmer before that, much as football coaches and basketball coaches and tennis coaches played the game (with rare exceptions) before becoming that coach. Managers are essentially coaches of our teams. Knowing what our teams and our people face, and what brings them together, is a challenge.

I mentioned the change from shutting out the world to inviting interruptions. A large percentage of programmers are introverts. As a manager, you need to learn extraversion skills. We need to learn people skills, empathy, and emotional intelligence to make people comfortable talking to us. We want our teams to talk to us, to let us know what’s important, what’s getting in their way, and how we can help.

How much training do managers usually have before they’re thrown into the fire?

Great question. I have talked to, by now, thousands of managers. … We’re looking at five percent or less of managers who have had a day of training. It’s ironic the amount of training we expect programmers to have compared to the lack of training that we expect of managers.

You also pointed out that many programming managers fall into the job almost without any planning, almost by accident.

Becoming a manager is not something you get to decide. You may think, “This is my [intended] career path.” You can train for it. You can prepare for it. But somebody else decides to give you an offer to be a manager. That offer could come because your manager quits, and then his manager says, “Ron, you’ve got some people skills. You manage the team.” That happens more often than not. You are stepping into a role that you haven’t given the thought that it deserves.

What should you think about before you accept that role?

They are very different roles, programming and managing. As a programmer, you’re working on code. You’re working on technical challenges. When you succeed, it’s so ecstatically wonderful that you jump out of your chair and scream with delight. At least I have!
You don’t get that as a manager. You don’t get the same excitement and joy coming from achieving. You also don’t get quite the same level of despa- rior right before that. But as a manager, you’ve got more breadth. You’ve got less say about any particular thing but more say in the game. You’ve got less ability to dive deep, but way more ability to dive broad.

It’s a fundamental decision about how you want to spend your time. Do you want to spend your time helping a whole group of people to get that joy, or do you want to be the one getting it?

For a programmer, that joy comes from, as you said, putting code out that works. What does success look like for a manager?

It’s less measurable. It’s when you feel that, “I’ve helped to enable a high-performance team. I’ve helped people to see what their contribution is to the bigger picture. I’ve helped people to see how they are delighting customers. I’ve helped people to have each other’s backs, to come together as a team, not just a group of individuals.” There’s a huge gratification that comes when you realize that.

In your book, you also show the archetype for what a bad manager looks like: the pointy-haired boss.

The pointy-haired boss comes out of Dilbert. It’s the boss no one listening to this show wants to be. We’re trying to be the boss [about] whom programmers say, “It was great working for Ron. If he ever calls me again to work on one of his teams, I’m going, because it was a peak experience.”

Is it a matter of avoiding bad things, is it something positive you have, or is it a combination of the two?

Absolutely a combination. Managers need to set boundaries without micromanaging, to set goals without telling people what to do and how to do it, to enable them to own their code, and to bring all of their thinking to create something better.

Can you describe what happens when a manager tells a programmer what to do and how to do it?

Micromanagement causes people to step back. If I’ve got somebody telling me what to do and how to do it, I’m going say, “I’ve done that. What do you want me to do next? And what do you want me to do after that?” I’m going to become an order taker instead of bringing my whole self to the product and to the team. It’s counterproductive.

Do you think that most of the time requirements, goals, or even values aren’t communicated clearly enough to developers? Is that the cause of many problems that we see?

Communication is one of the big problems that I run into. Software development is a team sport.

One of the big chasms is between product managers and the developers. Without being told the “how,” we need to understand as much of the “what” as possible. We need to understand the problems that our customers are trying to solve and how they’re struggling with [those problems]. Then we need to come up with options that our product managers [like].

Many organizations use Scrum or agile processes for communication between the team and product managers. One of your talks pointed out that the agile diagram doesn’t even depict the manager. That brings up the question, “If we’re agile, why do we need managers?”

[Agile teams are] cross-functional. They have Web developers, business logic developers, database developers, product managers, Scrum masters, testers, sometimes database administrators, and other functions. That team can deliver business value to customers. But what it can’t do is the care and feeding of [each specialty role].

The database developers need best practices. The database developers need to grow their careers. What managers do is to feed all of that and support all of that.

I’m not sure if [agile] is exactly a matrixed organization in the classic sense, but it’s definitely matrixed from the standpoint that the Web developers report to somebody who did Web development. [Their manager’s job] is not to tell them what to do but to support them in their growth and communication across their specialty.

You said in your book that there’s a distribution of skills in a team. Some programmers make outsized contributions, but that might not be enough to have a successful team and launch a successful product. Can a good manager take any group of individuals and turn it into a successful team?

There are categories of programmers. One category is jerks and bozos. When we’re building teams—and software development is fundamentally a team sport—bozos and jerks do not fit.

Can I take any group of people and make them into a team?
Probably not. Can I have a conversation with someone who’s acting like a jerk or about how we’re not going to have that on our team? [Where I] ask them, “Do you want to be on a team, or do you want to leave?” Yes, I have done that, because the team needs me to do that.

How do you build a successful team?

[We learned this from] a guy named David Vydra who is a guru in test-driven development. David has observed that during an interview, peer programming with a candidate for half an hour will save everyone a huge amount of time. During that time, we learn how the candidate programs, but we also learn how they collaborate and whether this is someone we want to work with. I think that’s a really valuable way of identifying people who will work and become part of our team.

Rick Sheridan, who runs Menlo Innovations, an agile-based consulting company out of Ann Arbor, Michigan, uses that to the max. They bring in 50 programming candidates at a time. They will pair them up with each other and have a third person watching [each pair]. The goal for the 50 people is to get their pair hired, not to get themselves hired. If you can make your pair look so good that we want to hire [the other programmer], we’re likely to want to hire you, too. Hiring managers have to think about who can work with the people on our team effectively.

Is that also a good measure of successful employees? They’re making their peers look good, not just making themselves look good?

Yeah, absolutely. Another measure of that would be, “How am I ensuring that my peers know what I know?” Every member of a team, even those just out of college, have unique expertise and experience. Our goal as managers is to foment the cross-pollination of that expertise and experience. Every member of a team should be thinking about, “What do I know that would be useful to my teammates to know, and how do I share it?”, and leveraging their manager to make sure that they are.

ABOUT THE AUTHOR

NATE BLACK is a software engineer at Sleeperbot. Contact him at nathanael.black@gmail.com.