BLOG@CACM

The *Communications* website, http://cacm.acm.org, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

twitter
Follow us on Twitter at http://twitter.com/blogCACM

DOI:10.1145/3542813 http://cacm.acm.org/blogs/blog-cacm

Advancing in the Technical Hierarchy

Doug Meil considers the steps to becoming a technical leader.

Doug Meil
Developing Technical Leaders
https://bit.ly/3x4aJiv
April 4, 2002

How does one get to Carnegie Hall? Practice, practice, practice, so the saying goes. How does one become a technical leader? Practice is certainly important, but at what? This post describes tiers of technical hierarchy with major topics to consider at each level.

Individual Contributor

This is the tier where everybody starts, and most people likely stay their entire careers. Staying an individual contributor is hardly a bad thing, it just depends on personal interest and career goals.

Individual contributors generally go through three phases: early-career/junior, mid-level, and senior, with the differentiating attributes of the latter phases being able to design, estimate, and implement from a "whole problem" perspective, the ability to look around corners, and mentor others. Experience is critical in being able to make this transition to greater engineering responsibility, but years of service is not the only determining factor. Some developers might have 15 years of experience, but might still be operating at a competent—but still junior—level if they don't continue growing.

Domain vs. Technical Specialization vs. Organization

A software engineer on Wall Street needs to learn about financial instruments. A data scientist working for a hospital needs to learn about healthcare. Those are *domain* skills and are important for functional delivery. As for technical skills, there are a plethora of topics to consider as new programming languages, frameworks, and techniques keep appearing. Technical professionals need to be competent in many areas, but it takes a lot of time to become an expert in something, and this is where focusing on a thing means *not* focusing on something else—or not sleeping—as there are only so many hours in the day. There are no "wrong" decisions on specialization, as long as those decisions line up with personal goals and interests.

There are some potentially mutually exclusive specialization scenarios to be aware of. For example, if an engineer chooses to specialize in Framework X in an organization and the organization decides to prefer competing Framework Y for future efforts, the engineer has a decision to make to either learn Framework Y, or find an organization that prefers Framework X. Or perhaps an engineer specializes in Domain X, but there are massive changes in that domain causing instability. The engineer now has choices to make in terms of whether to stick with a particular domain/industry or to make a transition to something new. These are not easy decisions and there are no pat answers.

Managing Up

Learning how to *"manage your manager"* is an important skill, and something that is not frequently taught. The traditional top-down way of thinking is that the manager initiates conversation and the employee responds, and employees that initiate conversation are either wasting their manager's time or are suck-ups. That viewpoint isn't helpful or in anyone's interest.

Individual contributors should periodically check in with their managers *seeking* feedback. Don't assume a manager is going to actively manage your career. Likewise, individual contributors need to state their needs to their manager: don't ask, don't get. There is a balancing act of not being *too* chatty, of course, but going radio silent on your manager until annual review time is worse.

How Long To Stay?

How long one should remain an individual contributor is a tough question. I would recommend not rushing up the ladder. Get as much experience hands-

on as possible, in as many products/projects as possible, in as many release cycles as possible. The experience—and scar tissue—with help down the road.

Technical Lead

"Technical lead" is often a position that people fall into; it has elements of a senior individual contributor, a large heaping of project manager, and "manager-lite" duties. It also is used as a testing ground for being a manager, both in terms of whether people are effective at the role, and whether they want to do it.

Being a technical lead is the hardest transition in leading people because it is the first time where one is required to *delegate* to be effective. A seven-person team that delivers successfully, but where the technical lead is doing 80% of the work, isn't sustainable. In that scenario, the technical lead might still be able to play the role of "hero" for a time, but that is bound to produce resentment in the rest of the team (such as, *"What are we doing on this project? Are we a fan club?"*), if not burning out the lead.

Managing Up and Down

The technical lead needs to be able to manage communication up to a manager, and down to the team. That means the *entire* team, not just a few people.

Leads Who Code

Technical leads typically want to stay hands-on, understandable given that is what they were previously doing as senior individual contributors. It's reasonable for a technical lead to have *some* committed deliverables for project or release, but the lead shouldn't overdo it.

Technical Reskilling

Getting back into full-time hands-on work can sometimes be a challenge, depending on how long one has been in this position, so plan ahead for refreshing technical skills being a tad painful. It's always surprising how quickly skills that are taken for granted can get rusty.

Manager

On top of responsibility of operational execution, roadmap prioritization, budgets, and customer/stakeholder escalations, there is all the other *"manager stuff,"* such as performance reviews, salary increases and promotions, overseeing growth plans for employees, hiring (for growth and replacement), expense approvals, staying on top of compliance and regulatory topics, chasing down approvals for all hardware and software needed by the team, as well as establishing relationships with the rest of the organization, to name a few. Having fun yet? Interested? Who wants in? Management is a lot of responsibility and doing it effectively is hard, but essential for organizational success.

Managing Up, Down, and Sideways

Managers also have the up and down communication challenge, but don't get to use the *"I'm just a technical lead"* excuse. People expect managers to know what's going on and to have answers, and if they don't, to go find them.

Managers Who Code

"Managers who code" is almost a trope in the technical community. It's a badge. It's a signal the manager still has technical chops. But there is a tremendous difference between a manager reviewing designs and code or participating in an occasional experimental project, and a manager who has committed deliverables on a real timeline. The latter could work under a few specific conditions, such as the team being relatively small, the team is seasoned, the product or framework being relatively stable, and the organization being relatively stable. It can happen.

If any of those are not present, a *"manager who codes"* is a huge red flag.

Technical Reskilling

Technical re-skilling for hands-on work after years in management is hard. It's not impossible, but certainly hard.

Director

Many people have the director title; even some individual contributors have the title. The traditional definition of a director is a *"manager of other managers,"* thus the responsibilities of a director are like those of a manager, plus more.

A director's most important tasks are stakeholder coordination, prioritization, resourcing, and clearing obstacles to delivery, in that order. While delivery is certainly important, it doesn't mean anything if it's not something people want, let alone when they need it.

Directors tend to live in a PowerPoint universe of planning and proposals, and it takes a lot of effort to stay grounded in reality, so watch out for that.

Managing Up, Down, and Sideways

Directors have all the communication responsibility of a manager, plus more. Effective directors need to be 360-degree communicators, and preferably information radiators.

Lastly, never underestimate the value of approachability. It's important at all levels, but especially important at this level as directors should be building alliances, agreements, and consensus, absence of which makes things very painful for the teams, and everybody suffers.

Directors Who Code

With the above caveats regarding what *"coding"* and *"director"* mean, this should never be a thing. Directors should be *directing*. If they are not, then something has gone horribly wrong.

Technical Reskilling

Never say never, I guess.

Other Food For Thought: People Join Companies, But Quit Teams And Leaders

Truer words have never been spoken.

Team subculture will have a greater influence on an individual's experience than any larger "corporate culture."

Keep Going

Healthy projects and products deliver, and keep delivering. Don't get stuck in a "perfect prioritization" or "reprioritization" rut. as it drives technical folks both nuts and out of organizations.

Further Reading

Management/Leadership Books That Technical Folks Should Read:
"Extreme Ownership," by Jocko Willink
"The Phoenix Project," by Kim et al.
"Great At Work," Morten Hansen

Relevant BLOG@CACM Posts:

Anna Karenina on Development: The importance of continued delivery.
What Happened to Watson Health?: A case study in portfolio management and what happens when a business doesn't have stakeholders and priorities aligned.

Doug Meil is a portfolio architect at Ontada. He also founded the Cleveland Big Data Meetup in 2010.