Career Hacks Careers

# Techie to tech lead: My five biggest mistakes

Peter Gillard-Moss

Published: Apr 24, 2018

As a young, ambitious developer with a strong sense of my own talent, I was eager to become a tech lead, and it took less than four years for me to achieve this goal. But over the next two years, the experience and reality of leading a team put me off leadership completely. For several years after, I retreated into the security of the technology, shunning any opportunity to take on more responsibility.

Over time, I gained an understanding of the root causes of my mistakes and eventually regained the confidence to accept new leadership opportunities and grow as a leader, and with the right support over the last two years, I've grown as a Head of Technology inside ThoughtWorks. As I have coached and mentored other leads, I've learned that some of my mistakes weren't unique to me but are common among technologists who move into leadership.

I'd like to share my mistakes and what I have learned in the hope that others may benefit.

## # 1: I believed my technical ability entitled me to lead

In my first tech lead role, I had hired a talented young graduate. We worked really well together and pushed each other forward into new ideas and skills. Within a few months, I realized that despite their inexperience, they were more technically able

than me. As the tech lead, I felt that I had to prove myself as the most technically capable; I'd feel insecure every time they demonstrated their ability to others (especially my managers), concerned that they'd view them as more suitable for my role than I was. What was originally a great working relationship suffered from my feelings of competitiveness and mistrust. Rather than seeing them as a collaborator (or even as a potential successor), I was seeing them as a threat.

Someone's technical ability is often one of the biggest factors when choosing them as leaders. It's also certainly the most visible factor. And here began my first mistake: conflating that ability with the entitlement to lead.


In my example, this conflation led to unhelpful competitive behaviours. A lot of my insecurity came from my environment and my mistaken sense of entitlement. But even in the most modest and deserving of people, this mistake can aggravate imposter syndrome, and the lead may perceive other members of the team as more worthy; they may underestimate their teamworkcontributions and struggle to understand their own worth. When this happens, leads can become indecisive and

understand their own worth. When this happens, leads can become indecisive and avoid making decisions, rather than trusting those they know to be capable.

This mistake prevents good people from putting themselves forward for lead roles, as they believe the most technically competent individual should be the lead.

## # 2: I focused on tech when I should have broadened my capabilities

I was very excited about my first lead role. It was a brand new team with a greenfield project. I introduced new technologies and practices: unit testing, ORM, continuous integration, pairing, optimistic source control, nant build files. If I read a blog post about an exciting new idea, I incorporated it into our processes and tools.

Choosing tools and technologies was iteration 0 work. Now came the day-to-day stuff. The managing, dealing with stakeholders, representing my team in meetings, performance reviews, recruitment, budgets, etc. And while I knew enough to select svn, nunit, CruiseControl, and C#, I had no idea about all that other stuff.

I began to resent the activities I didn't understand and found difficult. Rather than address my gaps, I focused more on the things I felt I could control: technology. This only aggravated my imposter syndrome, especially if I was in a meeting with other managers who'd confidently talk about growing their teams by bringing in training or making business cases for investment or new team members or disaster recovery plans, etc.

As previously mentioned, technical ability isn't the only basis for suitability for a leadership role. Alongside technical prowess are many other strengths: from influencing and persuading, to coaching, to understanding processes, to being a trusted advisor, to dealing with complexity, to strategic thinking (and many more). These competencies are much less visible and are probably needed in more nuanced ways throughout the working day and week.

It's important for new leads to understand the significance of these other competencies and the need to build themselves out in breadth across many areas.

Otherwise, they risk turning towards their most visible strength — technology — and focus on building depth.

Another trap is that it's relatively easy for a technologist to build up new technical competencies. We're used to acquiring technical skills; we know the formulas, the blog pages, the StackOverflow posts, the thought leaders to follow, the conferences to attend, the books to read. Given a half decent 'Hello World' example and a few spikes later we're well on our way to picking up a new technology.

But other competencies don't work this way.

> **"There isn't an obvious 'Hello World' for coaching or time management or influencing or persuading or articulating business value. "**

We don't know who to follow on Twitter to get the 'right way' to communicate decisions or organize teams or manage stakeholders.

This makes moving into the new domains required for leadership expensive. As many of these domains are relatively new, the lead is left feeling lost and intimidated. They may also mistakenly prioritize developing new technical competencies which are quick and easy to pick up and have obvious value over alien competencies, which are hard to get into without clear value.

## # 3: I continued to see myself as an individual producer

After a few months into my first tech lead role, I was becoming increasingly anxious. It seemed like I needed to do a lot of things that didn't feel productive. Less and less of my time was spent pairing and writing code as I was busy in meetings, replying to emails, answering questions from stakeholders, etc. I felt as if every story with my name on it wasn't progressing. My response was to sacrifice my personal time to make up for what I'd missed due to these distractions (as I perceived them). I quickly became overworked.

What I hadn't realized when I moved from a techie to tech lead, was how my responsibilities and metrics for success, in terms of being a producer, had changed. If you look at any delivery team, you can see the main goal is to maximize the amount of 'value' produced. In this light, there are two main activities in the team: those that directly produce value, and those that work to maximize the impact of what is produced. One of the simplest (but by no means the only) ways to do this is to ensure that the producers (in this case the developers) have the maximum time to focus on productive activities (i.e., producing code). The rest of the team, from PM to XD to BA to QA are working to ensure that the flow of work is as correct and smooth as possible and doesn't get blocked or create failure demand (due to bugs, etc.). Simple version: to keep developers productive, the team around them work to remove any obstacles or distractions so developers can focus on writing the right code to deliver value.

When someone moves from techie to tech lead, they're shifting from being one of the producers to being part of the team whose goal is to maximize overall impact (which includes removing obstacles and blockers).

This misunderstanding places a lot of stress on the new tech lead, especially around time management, specifically a feeling of not having enough time to code or pair with others.  Sometimes this is expressed in frustrations of too many meetings or a desire to have meetings only at certain times (core coding/pairing hours etc.). In some extreme cases, I've seen this manifest as a complete rejection of any activities which were not considered technical (talking to stakeholders for example).  All of these originate from the tech lead's perception of being a direct producer rather than as someone who is maximizing the impact of others.

In my case, it even led to me perceiving myself as the main, and an indispensable, contributor to the team. I've seen this happen a lot with others too, especially in smaller teams where there is a large skills gap between the lead and the other members. Under the combined pressure from additional responsibility along with normal delivery pressures, the tech lead can feel that if they shift away from writing code, the productivity of the team will significantly suffer.

## # 4: I wanted to know, and control, everything when I needed to empower others

One day, I was in a meeting when a manager asked about a bug. Having never worked in that area of the application I struggled to answer the question, and the more I tried, the less convincing I became. In the end, for fear of exposing my ignorance, I made an educated guess. It turned out that other teams started making changes in their applications to accommodate my guess and, after wasting a load of effort, they discovered I'd been wrong. I attributed my failure to ignorance and tried to prevent it happening again by reading the commit log at the start of every day.

When reading the commits, I noticed a piece of code that looked rather procedural and I felt could have been more object-oriented. I sat down with the developer and discussed trying a different approach. I decided to pair with them and became absorbed in this small area of the code for several days. At the same time, another pair were working on a significant architectural change. When they released, it caused a bug due to a simple incorrect assumption which I should have picked up.

I was honest with my manager about what had happened: I'd been focusing on

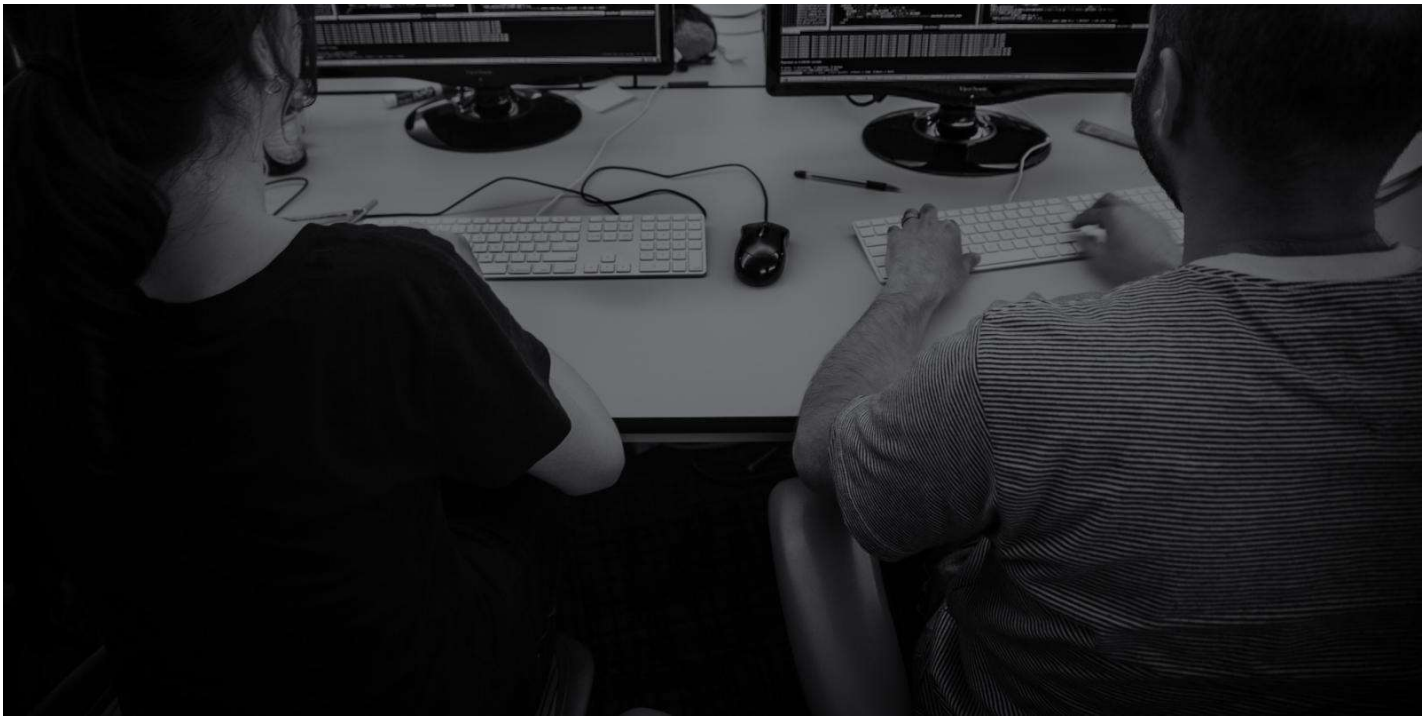
I was honest with my manager about what had happened, I'd been focusing on getting this code right and missed what the other pair were doing. But I struggled with the feedback that I was worrying about things that weren't important. Surely making sure we were doing OOP code was important? Their response? I needed to pick my battles.

Both these situations had one thing in common: that I couldn't be involved in everything that was happening. It wasn't humanly possible. As a tech lead, I had to trust others with responsibilities and focus on the most important things. Sometimes that would mean ignorance and other times it would mean that things weren't done exactly the way I would choose to do them, or even that they were done in a less than ideal way.  Saying "I don't know, I need to check with the team" was OK but accidentally spreading misinformation was harmful. And having little things that weren't quite perfect, like a bit of procedural code, was OK. But taking on large architectural change without my oversight was not acceptable.

I also realized that there are other tools, such as mentoring and running code reviews as a team, which are far more effective in improving the code base over the longer term than trying to fix every small issue, as and when it arose.

## # 5: I didn't recognize that the signals changed

I'd attend project meetings, suggest improvements, but my frustrations grew as I felt nothing was changing. My contributions didn't seem to turn into anything concrete. We'd discuss having the test teamwork with the developers on each story; we'd try it once and then we'd go back to queuing everything up for pre-prod. People would agree that the developers should own the database, but we'd still wait for the DBAs to build stored procedures. I'd coach developers on TDD but one iteration in, everyone had given up on it. Everything felt like a talking shop where nothing got done, and I felt increasingly frustrated and despondent.



When technology is your every minute of every day, you become attuned to the feedback loops and signals put in place. We're part of the story wall: our brains release endorphins every time a build passes or story goes to production, and cortisol, when the build breaks or we detect a story, is stuck 'in development'. But when you move into a leadership role, the signals come from other places.

It's hard for techies to detect this shift especially as the concerns of each day are mainly the same: stories still need to be picked up and delivered, the code still needs to be written. So the signals a new lead receives still look technical.

When moving into leadership roles, we need to look out for and learn the other signals. If we want to change direction, we must influence others. We must learn to see the signals that tell us who understands our ideas and whether they're convinced and supportive or skeptical.  There are also other signals, for instance team members struggling with the workload and needing support; or if they're missing the tools to develop new skills or disciplines; or there are gaps in communication structures leaving the team ignorant of what it's supposed to achieve. Often these signals are hard to detect because they come from people — who often hide or mask or highlight other things.

> **"Detecting these signals is almost a sixth sense which**

> **you develop with experience, and when you are not attuned to them, you don't even realize they exist."**

And they also have to be treated with caution: they may not always be caused by what you think.

The really good leaders seem to be highly attuned to all these signals and understand how to interpret and respond to them. Although it can look like a dark art, akin to reading the tea leaves or animal entrails, there are real rational approaches which, while not as deterministic and certain as technology, are learnable.

## Growing from my technical roots

One thing you may have noticed is that a lot of these mistakes relate to each other. For example, it's hard to stop seeing yourself as a producer when you believe your technical ability is your reason for leading. That makes it harder to develop other skills. The way these mistakes fed into each other is what put my leadership journey into a death spiral.

Taking a leadership role is a big change and can be very daunting, and when we, as humans, struggle, we turn to our strengths. And while technical ability needs to form the roots of leadership, we need to grow from our other strengths as well. Whether those are our abilities to work with people, to communicate, convince and persuade others or are technical skills which are transferable such as an ability to learn fast, pick up new things, reason about complex matters or reframe problems in more understandable terms by building abstractions. Once I felt confident in my strengths, I began to see the opportunities to grow new skills and branch out into new exciting areas of learning and development.

It's also crucial to understand that the leadership journey isn't linear.

> **"It's fine to play a leadership role and move back to pure**

> **tech. It will only make you a better technologist. And likewise, leadership is not exclusive to those who carry the label 'lead'."**

Teams should create smaller opportunities for development, without taking a lead role and find ways for others to practice developing the other skills in safer environments.

*Disclaimer: The statements and opinions expressed in this article are those of the author(s) and do not necessarily reflect the positions of ThoughtWorks.*



## Related blogs

Career Hacks

**What I Learned While Becoming a Tech Lead**

Jeff Norris

**Learn more** >