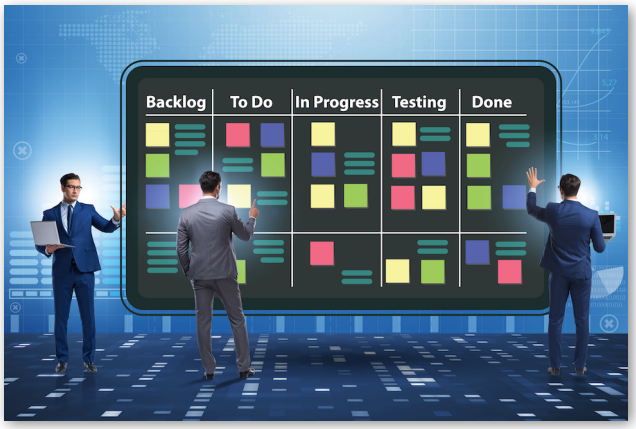
The 5 Most Common Mistakes in Software Leadership

Let us help you as you lead your team to success!

1 Unclear Priorities



"Tom, our sales guy just visited with one of our dream prospects this week. If we had feature X (not currently on the roadmap or backlog) we could land them."

"So - you want us to change everyone's priorities in case we can sway this dream client to consider us?"

There are so many variations on this theme.

- Maybe it's sales.
- Maybe it's senior management changing their plan.
- Maybe it's a public security breach that motivates a change, or
- Maybe it's just that the unprioritized backlog is so big that developers can almost randomly pick anything they want and claim that they are making progress.

The fact is that without a clear list of the things that are needed, you'll be wandering in the desert, and you won't be shipping "100% of something."

This "too many priorities" problem is often like the parable about the blind men describing different parts of the elephant, each believing they had a complete picture, but really only able to describe the part closest to them.

- Sales sees the gaps between what customers say they want and what the product does.
- Operations feels the pain of the app downtime and upgrades not working smoothly.
- Senior management wants to see the roadmap and BELIEVE it.

And you want to see your developers delivering business value and having all of you get credit for the hard work you're doing.

Next step:

Take an honest assessment of what your team is actually working on now.
Look seriously at the backlog, particularly the stories that have the most "noise" around them and estimate what it would take to deliver them.

Then have a conversation with key stakeholders from different areas about what trade offs can be made.

"If we deliver this for sales, it means that operations can't have that feature that lowers overtime for another quarter. Do we agree that it's ok to do that?"

When new "urgent" needs come up, take a look at what *won't* get done so that stakeholders understand more of the consequences of insisting their current hot item gets done.

Vague Requirements 2

Often, there is so much pressure to "make progress" that leaders drive their teams to begin working even before there is a clear plan for delivering business value. We call that "definition of ready" - when user stories, requirements or even features are clearly enough defined that they are ready for a skilled developer to make measurable progress.

When this is happening frequently, it may show up as:

- developers having trouble estimating work,
- and either a lot of rework because what was built is not what the customer wanted, severely delayed features, or developers just creating artificially high estimates based on uncertainty.

Next step:

Sit down with your developers and review some stories in the backlog.

Ask them to rank specific stories based on how confident they are in knowing exactly what is needed - start with high or low, and then look at recently shipped stories and compare what was built with what was documented in the original story.

You may gain some powerful insights about how "ready" your backlog actually is.

3 Communications

"Tom, it's been a while and I'm going to need you to send that information to me."

"I sent it several days ago!"

checks app
"Message Send Failure"

We've never had as many communication channels as we do today, and yet, how many times have you experienced a message not getting through?

This can leave us confused and disconnected - the exact opposite of the promise of the *"information at your fingertips"* that Bill Gates famously promised so many years ago.

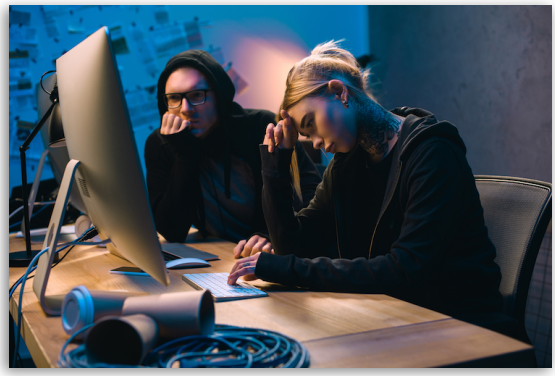
There are many causes of low communication, and tech tools are just one of them.

Next step:

Take some time and identify which tools your team should use (email, text, instant message like Slack or teams - including whether DM or channel)

- When should each tool be used? (e.g. email for "normal" and IM for "urgent"?)
- For each tool, what is the expected response time? Within a day? A few hours? A week?
- When do you expect team members to be accessing the tool? (e.g. Off hours, do team members need to check and respond to messages?)

Write down your list and then set aside some time to discuss expectations with your team!



Trust Issues 4

Google has done extensive research on high performing teams in their Project Oxygen and Project Aristotle.

One of their key findings is that teams who significantly outperform "regular" teams have one characteristic that sets them apart more than any other.

Experts call that "Psychological Safety" - it basically means that team members feel safe to be themselves and to take appropriate risks.

How would your team respond to these statements?

How strongly do you agree or disagree?

1. If you make a mistake on this team, it is often held against you.
2. Members on this team are able to bring up problems and tough issues
3. People on this team sometimes reject others for being different.
4. It is safe to take a risk on this team

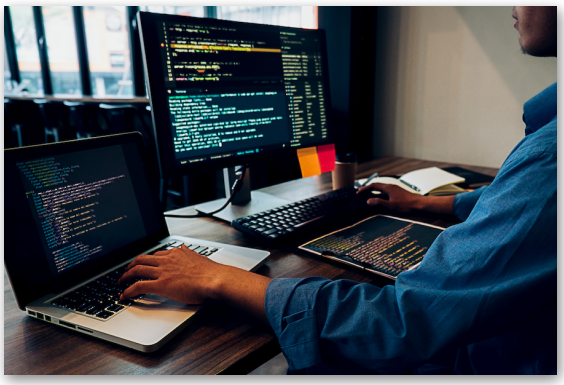
Next step:

Your goal is NOT to get the "right" response -
Your goal is to to increase your team's shared belief that the team is safe for interpersonal risk taking

Three research-based things you can do are:

1. Frame the work as a learning problem, not an execution problem
2. Acknowledge your own imperfection and fallibility
3. Model curiosity and ask a lot of questions that allow your team members to share their point of view.

5 Developers Picking Your Tech



"Tom, I know I'm late with this feature, but I've been having issues getting this to work in MS Distributed Transaction Coordinator"

"I know I'm not the architect here, and you're definitely more skilled than I am, but our system is not distributed or transactional. Why are you using that tool?"

Dev engages in the "cloud of confusion"
(Hoping I'll be overwhelmed with a barrage of buzzwords and forget my question.)

After several back-and-forth exchanges
"Well, I picked that tool because I thought it would be cool."

This is an old story, but i've worked with leaders like you who see it over and over again. Developers get a little bored with the everyday stuff. Maybe they are afraid they will be left behind, or maybe they feel like access to that new feature will be super helpful.

A lot of times this works out fine, but sometimes it's a long path to an expensive thing to maintain. Like the team I worked with recently who had built their whole system in Java, but then had a developer code a bunch of components in .NET because he was available and it was what HE wanted to use.

He was optimizing for his preference, but what happens when that dev leaves the team (as happened a few months later?) Who will support that part of the code?

Next step:

Write down and share the key things the company is optimizing for - including shared knowledge, vendor/community support for the language, support lifecycle, alignment with company standards, total cost of ownership (including licensing plus maintaining knowledge about it,) scalability, and other factors.

As decisions are being made for building a new component, review the proposed solution compared with your prioritized list, and if the one that your developer loves, doesn't match up, help them compare their criteria with the company's criteria.

Now what?

What will happen next?

Maybe you see your team in this situation, what happens if you keep going the way you are going?

Your team is working hard, but not getting credit, and you know that you all are not delivering as much value as you know you could deliver.

Without some kind of change, things are not going to get better.

You're still going to have that empty feeling in the pit of your stomach when you see the caller ID, the email from your boss or other company leaders, and no matter how hard you push yourself and your team, it's likely to be frustrating.

But it doesn't have to be this way.

Imagine a time when

- Your leaders agree on an achievable roadmap.
- Where realistic tradeoffs are made and
- Your backlog is focused on the highest value items first.

Imagine being in the room full of stakeholders who all are saying

"We know it's been tough and we appreciate that you and your team delivered as much as they have this year!"

I'd love to help you on your journey.

Set up a call with me today. Whether we are talking about a program assessment, a workshop to get all team members on the same page, team building, team support or individual coaching, together we can help you be the hero your company needs in a software leader.

Next Step:

Set up your call now!
https://calendly.com/brighthillgroup/quick-call-with-tom

Schedule a call