Engineer

Technical leadership

setting direction and sharing technical decisions

What it is

Technical leadership is the work of helping a team make, explain and follow through on technical decisions. It includes making the goal clear, giving people room to use their judgment and creating a way to learn from what happens after a decision. It can be part of a manager's job, a staff engineer's work or the responsibility of someone leading a single change. The test is whether the team can make a better decision and carry it through with clear ownership.

A decision about architecture has several owners in different senses. The person making the decision needs a mandate and a deadline. The people implementing it can see costs that are invisible on a diagram. Whoever will operate the result has to be able to diagnose and recover it. A product owner or sponsor can say which user outcomes matter and which constraints are fixed. Technical leadership gives all these facts a place in the decision and makes clear who closes it. Asking for input creates an opportunity to identify a risk. It does not give every participant a veto.

The joint ACM and IEEE Computer Society software engineering code gives this authority a concrete purpose. Principle 5 makes engineering managers and leaders responsible for promoting an ethical approach to managing development and maintenance. Its clauses cover procedures for quality and risk reduction, realistic estimates and fair consideration of concerns. Principle 7 asks engineers to help colleagues develop, review their work objectively and hear their opinions and complaints. These responsibilities fit together. Giving someone ownership includes giving them the information, support and chance to speak up that they need to exercise it.

Use the concept in a decision

Put “What is technical leadership?” to work in a real decision. We can help relate the concept to the system you are building or buying. Somebody on your side of the table when a supplier, a budget or an architecture decision needs a second informed opinion.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. One infrastructure plan for 14 departments — 14 research departments on one infrastructure plan

One infrastructure plan for 14 departments Get in touch →

Why it matters

Software work often brings incompatible constraints into the same room. A team can spend a week reducing the risk of a change and still miss a delivery deadline. It can meet that deadline and give support a manual procedure it has no capacity to perform. Choosing among those costs is part of the work. The leader makes the effect on each party visible while there is time to change the plan. An early discussion about how to reverse a release can change the order of the implementation and avoid days of rework.

Delegation gives people closer to the problem a chance to make choices with better information. It also develops the next person who can lead work independently. A useful delegation defines the result, names the boundaries and agrees when to check in. A person who owns a migration might choose the order of the tables and the size of the batches. A decision that changes the promised outage or makes a rollback impossible could require another approval. These boundaries let the person proceed with confidence and let other teams see which decisions affect them.

Feedback closes the loop. During review, useful feedback names the observed behavior, who it affects and what a better outcome would look like. After the change, feedback compares the result with the original intent. A faster import may reduce overnight processing and still increase daytime contention. Reviewing both results helps the team keep what worked and reconsider what it originally assumed. A decision record earns its place when a new colleague can read it and tell which evidence would justify revisiting the choice.

What it looks like in practice

Imagine a booking team whose support staff sometimes have to correct duplicate confirmations. A meeting starts with proposals to replace the queue, add a lock or change the order of writes. The technical lead asks for one incident reconstructed from an order identifier, the intended behavior and a way to demonstrate that behavior. The group agrees on a rule the business can recognize. Resending the same booking request must return the existing confirmation, and a retry must never reserve a second place.

The engineer who knows the booking path proposes where to enforce that rule. A second engineer writes concurrent test requests and checks the result from both the customer and operations sides. Support tests the message it would use when a booking is retried. The lead makes the review deadline and owner explicit, and confirms with operations how the change can be paused if errors rise. The engineers choose the implementation. The team has a clear point at which to raise a change to the customer contract, with commercial and operational colleagues present for that decision.

A useful decision record might fit on half a page. It identifies duplicate bookings as the problem, records why the team chose an approach, lists the behavior demonstrated by tests and states what operators will monitor. It names who will judge the first week of results and what signal warrants another look. It also records a rejected alternative with the reason. If retries become much more frequent than expected, the next team can revisit the design without guessing why it was first chosen.

Engineer's case on setting a platform's founding technical decisions is a concrete example of architectural choices documented as part of the work. Its case on directing full-stack GIS map development gives another context, where technical direction extends across layers of a map product. The Technical Leadership & Consulting service describes the sort of engagement in which technical direction and assessment are part of the work. The booking scenario here is an illustration of the decision process.

Where it goes wrong

The lead becomes the approval queue. The same person must approve familiar changes, minor implementation details and work with genuinely expensive consequences. Waiting encourages people to combine unrelated choices into a

single approval. Agree in advance on which decisions are local, which need a specialist and which have consequences broad enough for explicit approval. Review how long each kind of decision is waiting before adding another step.

An assignment comes without authority. A developer is asked to own a result while another person keeps choosing the means in private. Clarify who decides, where someone may challenge the decision and what information needs to be brought to an escalation. If the constraints leave one permissible option, be direct about that before asking the engineer to take the work.

Feedback becomes a verdict on the person. Comments such as “you always overcomplicate this” give the author little to do. Identify one decision and show the consequence. “Two jobs can now update the same booking, so can we show what happens on a retry?” is specific enough for someone to reproduce. In review, immediate corrective feedback can sit alongside a separate coaching conversation about a broader pattern.

The decision has no return date. Teams remember who approved a design and forget which evidence would overturn it. Put a date on the first review or name an event that will trigger it. At that point, look at outcomes, unresolved risks and the effect on the people who use or operate the system. Keep the line of authority clear so someone can act on what the team learns.

Further reading

- Software Engineering Code of Ethics and Professional Practice — the joint ACM and IEEE Computer Society code, especially its principles on management and colleagues.
- Kubernetes community membership — an open project’s definitions of reviewer, approver and related responsibilities.
- Google’s standard of code review — review judgments, mentoring and improving the health of the code over time.



Glossary

Choose for your actual constraints

Bring the decision, the people affected and the options already considered. We can make the tradeoffs explicit and leave a documented recommendation.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is technical leadership?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/technical-leadership/