CORALINE ADA EHMKE

December 07, 2020 - Chicago, IL

Tweet this post! Post on LinkedIn

Engineering organizations of all sizes make technical decisions every day. Most of the time, these decisions are on a relatively small scale: an individual engineer or a small team solving a problem in the way that makes the most sense to them.

However, there are some decisions that have a much wider impact, and it's important to have a process for capturing those decisions. Such decisions contribute to a deeper vision of how your stack or toolchain is evolving in response to the changing technology and business landscape.

This can be challenging in a remote environment, as many of the socialization channels that work for onsite engineers aren't effective when your teams are distributed across the country or even around the globe. In a world of asynchronous communication, it's more important than ever to create inclusive and remote-friendly collaboration, decision-making, alignment, and documentation processes.

One way to meet the documentation needs and adoption goals of impactful technology changes is through Architecture Decision Records (ADRs). Adopting an ADR practice means creating a process for proposing and getting alignment on far-reaching technical decisions.

Here at Stitch Fix, we've been iterating on our own ADR practice for over 3 years, and we've learned a lot of lessons along the way.

a record that provides context on the "what", the "why" and the "how" of a significant technical decision.

To get a clearer picture of how an ADR communicates these points, we can start by exploring its overall structure.

Problem Statement or Context

The ADR opens by painting a succinct statement of the problem at hand. This might be as simple as a clear and concise problem statement, but could incorporate historical elements or communicate a desired future state (such as a larger goal for a core architectural improvement that you are building toward). This introductory section frames the problem space and lays the foundation for the rest of the ADR.

History

Most organizations have adopted ways of working, particular tool chains, or "blessed" sets of patterns to solve their engineering problems. It's important to reflect on historical solutions, to acknowledge their strengths and weaknesses, and to prepare for making the case for the...

Future State

What alternatives did we explore?

When you share an ADR for feedback, other engineers are likely to take inspiration from what you've written and may have their own ideas on how to solve the stated problem. Their feedback can be very helpful as you solidify the document, but sometimes the suggestions you get may relate to something you've already evaluated and eliminated. This section is good for "showing your work" and addressing some of those suggestions in advance, documenting the advantages and disadvantages of those alternatives, and explaining why you selected the alternative that you did.

Decision

The decision section of the ADR resolves the questions raised under the Alternatives heading. It is a clear statement of what the author has settled on as the most appropriate solution to the problem under consideration.

Example code

Since the ADR should reflect existing work that will be emulated throughout the organization's code base, it's helpful to provide sample code or even link to file diffs in a project repository. Pointing to such live code examples can help clarify your solution for people who learn best by reading code.

they need to make a change to their way of working, or their toolchain? How can they get started? In short, what do teams need to do to fulfill the requirements of the ADR?

Different kinds of ADRs and their goals

Not all ADRs serve the same purpose, or carry the same weight. There are different kinds of ADRs, some of which are highlighted below.

Best practice ADRs

Some ADRs reflect best practices that are intended to be broadly adopted. This might include aspects of service design, canonical ways of integrating observability or tracing tools, or detailed guidance on API protocols.

Whatever the specific practice might be, the goal of this kind of ADR makes design and execution more consistent, to increase the coherence and quality of code produced by engineering teams.

Novel art ADRs

effective at solving a particular challenge, in a creative way, and in a production setting. Novel art ADRs can improve productivity by making sure that engineers don't solve the same problem twice in different ways.

Directive ADRs

Some ADRs communicate organization-wide directives from technology leadership. Examples include the adoption of a new framework, a new language, or a fundamental infrastructure change.

Such ADRs should be labeled clearly so that anyone who reads it understands that compliance with the ADR is not optional.

Gathering feedback and getting buy-in

Authors should solicit feedback on their ADR at several points during the writing process. Remote engineers thrive on asynchronous communication, so keep that in mind when you are designing your review and feedback processes.

Feedback from Subject Matter Experts

There may be people within your organization who have particular expertise in the subject area of your ADR. You can share the main idea and an outline of your

Once you have your first draft completed, feel free to share it with your team. You may want to do this in a separate setting from the ADRs' final home (whatever you're using as an ADR repository). It may be helpful to create a shared Google or Dropbox Paper document. However you make it available, make sure your teammates can leave comments on the document.

Small group discussions

To ensure that everyone (including people who may be shy about speaking up in a large meeting) has the opportunity to ask questions and provide feedback on ADRs that are up for review, we conduct small-group discussions facilitated by senior engineers. Participants read the proposed ADR in advance, and the facilitator prompts everyone to share their thoughts and opinions. The facilitators are responsible for recording any concerns that are raised or questions that need to be addressed. They then bring these questions or concerns to the discussion of the ADR in a wider public forum.

Engineering-wide presentation, feedback, and alignment

Once an ADR has gone through these preliminary review processes, it's time to solicit feedback from the wider engineering team. At Stitch Fix, we do this through a monthly all-remote meeting, open to the entire engineering organization, that we call Forward the Foundation.

briefly about what problem is being solved, how the ADR proposes solving it, and other high-level details. Then we open the floor to discussion. We start by asking the small-group facilitators to share what came out of their discussions, then solicit feedback from other meeting participants.

One of the primary goals of a meeting like this is to get feedback and alignment on the ADRs that are being presented. Adoption decisions may remain outside the scope of the meeting, since approval is likely the responsibility of senior engineering leaders at the appropriate level (based on the scope, associated costs, or scale of the proposed change or innovation.)

However, we feel that it is important to revise and even revisit the ADR as necessary in response to the alignment discussions at Forward the Foundation. Although ADRs are not be subject to a "vote" by participants in the meeting, successful adoption requires general alignment on both the problem statement and the proposed solution.

Alignment is the key term here. As your engineering organization grows, driving to consensus can be difficult or even impossible. We prefer to frame the ADR presentations and any follow-on discussions as a process for refining details and understanding impact rather than challenging the core premise.

After approval

Once the ADR is presented and approved, it should find a home in a searchable, centralized, easily accessible location. You might add a section to your engineering wiki, create a dedicated repo on GitHub, or set up an easy-to-find Google Drive folder for all of your ADRs.

across the organization and visibility into what other groups of engineers are working on.

Try to make the feedback gathering and presentation phase of ADR adoption as smooth and friction-free as possible. If need be, write out the process and add it to the README or How-To section of your ADR repository.

Finally, set up a regular cadence for your ADR socialization and review meetings. People are often motivated by a deadline, so communicate timelines clearly.

Impact on growing teams

One of the primary benefits of an effective ADR practice is that it can help unlock and disseminate knowledge that might otherwise be siloed. As your engineering organization grows, even the most social developers can only build relationships with a smaller and smaller percentage of the overall population of engineers. This means that it gets harder to communicate across team boundaries.

ADRs can play a vital role in increasing cross-team collaboration. It often happens that multiple engineers or engineering teams will face problems similar to those your own team is taking on, or perhaps the organization as a whole needs to settle on a direction for an important technical consideration. Collaboration between teams on the technical aspects of the ADR is a great way to cross-pollinate ideas between teams.

Your ADR repository or wiki is also a valuable tool for onboarding new engineers, especially in a remote culture.

probably see how ADRs could be an important step in the evolution of your technology solutions and practices. ADRs help with the socialization of technical innovations arising from successful experiments, unlocking and inspiring engineers across the organization.

Establishing an ADR practice at your company can help your engineers more confidently deliver business value, in ways that are consistent with your organization’s overall architectural principles and values.

MultiThreaded