RFC: Introducing RFCs

Oskar Wickström

January 8, 2020

Status

PROPOSED

Abstract

This Request for Change (RFC) describes how we should communicate and give feedback on designs using RFCs in the distributed ACME team.

Introduction

In the beginning, there was Alice. And then came Bob. They exchanged phone calls and frequent short messages, quickly reaching shared goals and designs for everything between user interfaces and technical architecture. Carol joined, and the team needed to meet physically in order to have efficient design discussions together.

Since then, the ACME team has been growing even more. At the end of January 2020, including full-time and part-time team members, we'll have a team size of N. One-to-one communication will not scale when it comes to communicating and giving feedback on our designs.

The term *design* includes everything from business decisions, technical architecture, graphics, copy, and user experience. Everything we do is design, and it's cross-cutting throughout our product. Business decisions affect the database schemata, user interfaces put demands on cloud deployments and data processing, and so on. We should involve our peers early in our designs, and reduce the effort to give feedback and make adjustments.

The goal of this RFC is to enable *asynchronous* and *persistent* communication and feedback for our distributed team. It's explicitly *not a goal* to

weigh us down with heavy processes, or to demand formal specifications for our work.

Proposed solution

We'll use lightweight design documents, inspired by and named after the internet standards RFCs. This document is itself written as such an RFC, serving as an example. It's the meta-RFC, the fix-point of RFCs. Everything must start with recursion.

Our RFCs are meant to be informal and accessible, not academic or formal specifications. They should be fun to read, not boring. Even if a reader doesn't understand all technical or business aspects of an RFC, they should be able to read the abstract and introduction and get a general sense of its value.

It's up to the team to decide on the level of detail required for each individual RFC. If lengthy detailed discussion of a topic is needed, they can include one, or peers giving feedback can request one.

RFCs should be loosely based on the following structure, where the top-level items are top-level headings:

- Status, one of:
  - PROPOSED
  - REJECTED
  - ACCEPTED
- Abstract
  - A few paragraphs describing the RFC at a high level
- Introduction
  - Context
  - Problem statement
  - Goals and non-goals
- Proposed solution
  - Implementation
    - How it works
    - Pros and cons

  - Impact, for example on:
    * End users
    * Our team (ops, on-call)
    * Costs
    * Performance
    * Security
  - Quality assurance
    * Testing
    * Operational concerns
      · Deployment
      · Monitoring

- Alternatives
  - Alternative implementations, also including:
    * Implementation details
    * Pros & cons

Items in the structure above that are not applicable can be excluded. In addition to this structure, feel free and highly encouraged to include graphics and drawings, links, references, and whatever else helps convey your design.

Format

We use pages on the ACME space on Confluence to publish RFCs. Create a blank page under the RFCs page. The stage is yours.

To give feedback, use either comments below the page, or even better, inline comments. To add an inline comment, highlight the text you want to comment on and place your mouse cursor over that text for about 1 second. A button should pop up.

Risks

There's a risk that writing and commenting on RFCs will become burden and weigh us down. That it feels like heavy process that no one will have time for or feel encouraged to do. We have to reflect, and possibly adjust it, continously so that this does not happen.

Impact

As a side-effect of using RFCs for our designs, we'll have a log of the decisions and trade-offs we've made during the project. This is a a great start for new team members to read about the project and understand its history and current situation.

Designing together and getting feedback from team members with other perspectives will produce better designs. It will spread knowledge more efficiently, and enable us to work remotely and to respond when time permits.

Alternatives

The following could be venues on which we discuss design and reach decisions:

Slack

While being our preferred day-to-day, quick, and easy communication channel, Slack is not an ideal place to store long-lived documents and decisions. If you write an RFC, do post a link on Slack, though!

Email

Email is disqualified mainly on the same grounds as Slack. Its upside would be easily giving access to, and looping in, people outside our project. One can always export the Confluence page as a PDF and send to outside collaborators, or invite them with read permissions.

Meetings

While being called for in some situations, this should not be our primary way of communication. Meetings require time synchronization and co-location, and are generally a bad format for fair discussion, with social dynamics distorting the process.