caitiem.com

caitiem.com

Design Docs, Markdown, and Git

About a year ago my software engineering team, the Azure Sphere Security Services (AS3) team, found ourselves struggling with our design document process. So we ran an experiment, moving all our design documents to be written in Markdown, checked into Git, and reviewed via a pull request (PR). The experiment has been incredibly successful, so we've iterated and refined it, and have even expanded it to the broader Azure Sphere team. The goal of this blog post is to share our process and what we learned along the way.

Our original design doc process involved writing a Microsoft Word document and sharing it via SharePoint. Feedback was gathered via in person reviews, doc comments, and emails. Approval was then done over email. To signal that a document was the "approved plan of record" versus "an under review draft", we toggled a property on the document. Users could filter documents on the SharePoint by this property to disambiguate between the two states.

This worked fine when we were a small team, with a small number of documents, but became challenging as the team grew. For context the Azure Sphere team, started out as a handful of people working in Microsoft Research and has grown rapidly over the past 3 years as we've went from research project to Generally Available Product.

Challenges

Some specific challenges were identified via

the AS3 retrospective process. When evaluating new options we kept these pain points in mind:

- **Comments:** Understanding when comments were resolved and when forward progress on the document could be made was challenging. Tracking Comments in the Word Document, in person reviews, and across emails became cumbersome. It also was unclear when comments were resolved and what the resolution was. Finally once a doc was approved, all the comments were hidden, and this often lost valuable context.
- **Approval Process**: It was often unclear how to get approval or who had approved a document. The Word Document did not contain a section on reviewers and approvers. As the team grew there was some ambiguity on how the Approval process worked. In addition many approved design documents did not have the property set to approved, this resulted in the SharePoint becoming a mixture of approved, under review, and abandoned documents and it was unclear which state they were in.
- **Context Switching:** For the Individual Contributor (IC) engineers switching contexts and using a different tool chain, Word & SharePoint, was a barrier to writing and reviewing design docs. It added just enough friction, that design docs felt like a bigger challenge than they needed to.
- **Versioning:** Word and SharePoint do not provide a way to easily version a document. As the team and the product grew, there was a need to update and version documents.

The Experiment

To address some of these challenges the AS3 team began writing design documents in Markdown and checking them into a new EngineeringDocs Git repo in Azure DevOps (ADO). Reviews are conducted via pull requests by adding comments, pushing changes, and then resolving comments. Approval was given by signing off on a pull request, anything in master is considered the approved plan of record. Versioning was also greatly simplified as anyone could submit a pull request to update the document.

Single Repo vs Next to Code

One of the first early decisions we made was where in the codebase design documents should live. We discussed two options

- **Single Repo**: Create a new Engineering Docs Repo where all design documents would be checked in.
- **Same Repo as Code**: Design Docs should be checked into the repo for the code they implement.

We chose to use a Single Repo for several reasons:

- **Discoverability**: a downside of having the design docs living next to the code is finding all the existing design docs became challenging. With a quickly growing team we wanted to prioritize discoverability of design decisions to help onboard new team members quickly.
- **Large Designs**: For designs that didn't neatly map to a single service or library it was ambiguous where these design docs should live. For example where do design docs that span multiple

design? does that span multiple microservices live?
- **Unconstrained Design:** If an author first had to pick the code repository where the design doc would live, this initial choice would artificially constrain the design, to only consider changes to that part of the code base. By having the design docs checked into a single docs repository this artificial constraint was eliminated from the early design process. This frees up the design document author to think about the best way to implement their feature across the system as a whole.

Conducting a Design Review

The Azure Sphere team uses the OARP model for making decisions, so the below section describes approval and stakeholders in this context. I recommend having a well defined decision making process and integrating whatever that is for your team into the design document process.

Identify Reviewers and Approvers via a Pull Request

The first step in our Design process is identifying the stakeholders. The first pull request includes the title of the Design Doc and a table listing the OARP assignments for this document. The pull request author is always the Owner.

This serves a few purposes:

- It ensures how the decision is being made is clear.
- Informs the team that this is a problem we are attempting to solve now.
- Gives people the chance to opt out. If you are listed as an Approver or a Reviewer, and would like to delegate your responsibility or opt out of the

process you can.
- Gives people the chance to opt in. By notifying the broader team via pull request, teammates not initially listed in the OARP model can request to be included in the design process if they are interested, or feel they have expertise to add. This prevents people from feeling excluded or coming in late to the review process with a lot of additional feedback.

Once the stakeholders are all identified, the Approver approves the pull requests, and the Owner checks in the pull request.

Writing the Design Document

To author the design document the owner creates a new branch modifying the checked in shell document. It is highly recommended that input from Reviewers and Approvers is informally gathered prior to writing the document. This can be via white board session, chats, hallway conversations, etc... This ensures that the design review process is more collaborative, and there are few surprises during the formal review process.

Design docs are written in Markdown. Architectural diagrams are added to the design doc by checking in images or using Mermaid. The AS3 team often generates architectural images using Microsoft Visio. It is highly recommended that these Visio diagrams are checked in as well for ease in modifying later.

Once the design doc is ready for review, the engineer submits a new pull request. All members of the OARP model are listed as reviewers on the pull request.

Design Pull Request

Once the pull request has been submitted,

design review stakeholders can read and submit feedback via comments on the pull request. All comments must be addressed and marked as either resolved via document updates or won't fix.

The document can be committed to master once the Approver has approved the pull request. This design is now considered a plan of record.

Design Review Meeting

Design review meetings are not required but often held. A meeting invite is sent out ahead of time. Owners, Approvers and Reviewers are considered Required attendees, Participants are considered optional.

The meeting invite should be updated with a link to the pull request for the design doc to be reviewed, at least one business day prior to the meeting. The first 10-15 minutes of the meeting are set aside for folks to read the document and add comments to the pull request if they have not done so already. In either scenario feedback is added via comments on the pull request.

We provide two ways for folks to review the document, ahead of time or in the meeting to accommodate multiple working styles on the team. So folks prefer to digest and think about a design document for a while before providing feedback, others are more comfortable providing feedback on the spot.

After the reading period the design review meeting spends time focusing and discussing the comments. The owner takes notes and records the in room decisions in the pull request comments.

Updating the Design

Throughout the course of the project design docs may need to be updated. This can happen after design if a major change was made in implementation, or could be later in the life of the project as a new feature or requirement requires a modification.

Updating the design doc, follows a very similar process. A pull request with proposed changes are submitted. The original Owner and Approver should be considered required reviewers.

Conclusion

The AS3 team considers the experiment incredibly successful so much so that the broader Azure Sphere team has begun adopting it, including the Program Managers.

To summarize all the challenges we experienced with Word Documents and SharePoint were addressed by using Git and Markdown.

- **Comments:** Adding comments via the pull request makes it really clear which ones have been addressed, and which ones are still open. In addition the resolution of the comment is clear, either Resolved or Won't Fix. The comments and discussion in them are also not lost once the pull request is submitted, as you can always go back and look at the history of the document.
- **Approval Process**: By having an initial pull request identifying the stakeholders, how we are making the decision and who is involved is incredibly clear. In addition the participants are durably recorded, as is the act of signing off, approving the pull request.
- **Context Switching:** IC engineers no longer have to switch tool chains to

participate in the design process. What reviews code or design they need to do are easy to find and discover.
- **Versioning:** Versioning documents via Git and pull requests is incredibly easy. So much so that engineers often go and make updates to the document once they finish implementing the feature. In addition, having the history of the document has been incredibly valuable.

By utilizing a toolchain that developers already use day to day the process feels more lightweight, and writing design documents feels like a less arduous process. The Program Management team has also been incredibly receptive to using Markdown and Git. While these are new tools for some of them, they've embraced our growth mindset culture and dove right in.

One of the biggest benefits I've observed is the clarity it has brought to how decisions are made, and durably recording when things are done. On a fast growing team like Azure Sphere having clarity and durable communication are key to successfully scaling the business and the team.

Tech Insights

Previous post Next post

Leave a Reply

Your email address will not be published.
Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment