Go Proposal Process: Scaling Discussions

Go Proposals, Part 4

Russ Cox

August 22, 2019

research.swtch.com/proposals-discuss

[*I've been thinking a lot recently about the Go proposal process, which is the way we propose, discuss, and decide changes to Go itself. Like nearly everything about Go, the proposal process is an experiment, so it makes sense to reflect on what we've learned and try to improve it. This post is the fourth in a series of posts about what works well and, more importantly, what we might want to change.*]

The current proposal process refers to "discussion on the issue tracker" and, in the case of a design doc, a "final discussion," but we've never written more about how to make those discussions effective. A discussion of a small is typically a dozen messages or fewer; those are easy. Discussion of large changes is more difficult and does not always work well.

Scaling One Discussion

As a forum for active discussion of large changes, the GitHub issue tracker has serious flaws. As I write this, the "try" issue takes seven seconds to load, and when it does it reports that there are 798 comments but only displays the first 29, then a marker for "773 hidden items," and then the last 25. Hiding most of the discussion makes it impossible to search the page, which leads to people making points that have already been said. (For comparison, my outdated, manually curated summary page loads and displays 611 comments in 300ms. The problem is not the size of the raw data.) Using a discussion forum that displayed the entire discussion would be a step in the right direction. After that, it would likely help to display a threaded tree of messages, both to make clear what a reply is replying to and also to allow skipping over subthreads that are uninteresting for one reason or another. After that, it would likely help to add comment ranking that affects display order, to help surface the most important comments. One idea raised was to use /r/golang or a new subreddit for the large proposal discussions, and that seems worth considering further.

At the contributor summit, we asked that each discussion have someone serve in the role of "facilitator." The facilitator tries to point out possible misunderstandings as they happen, makes sure that a few people don't dominate the conversation, and tries to bring quiet people into the conversation. One point raised was whether it would make sense for the online proposal discussion to have a clear facilitator whose job is to keep a summary or decision document up-to-date as well as to keep the discussion on topic and non-repeating as much as possible. (In the first few days of a particularly active discussion, this might approach a full-time job.)

While better software and better process would help manage a large discussion, though, there is a point of diminishing returns, and we shouldn't focus on this one discussion to the exclusion of what precedes it. It is possible that instead of trying to scale one discussion we should scale by having many discussions.

Scaling with Many Discussions

As I mentioned in the large changes post, one problem with "try" was simply that it should have been a second design draft, not a proposal. Making it a proposal with a timeline made everyone feel like they had to rush to comment before a decision was made. In the more relaxed setting of evaluating a design draft, the more distributed "post a thoughtful experience report somewhere and link it to the wiki" scales much better and seems to be working well.

Requiring large changes to start with a series of design drafts creates space for a variety of different conversations about the designs, in different forums and media. Any important points discovered in those conversations can be reported back to the proposal issue and influence future drafts, without having to put every comment on the proposal issue. These various discussions would also help impacted users get up to speed on the details of the proposal, again without having every comment on the issue itself.

Overall, having many discussions would in turn reduce the criticality of the issue tracker discussion itself and therefore the demands the discussion places on the discussion forum, whether that's GitHub, Reddit, or something else. (As one data point, the Go modules issue received only 242 comments, compared to the current 798 for "try", quite possibly because there had already been so much community discussion in other forums before the issue was filed.)

Scaling with Offline Discussions

Another fascinating idea raised at the contributor summit is to make use of the Go Meetup network in the discussions of the most important, largest changes, such as generics. The idea would be to prepare materials to help meetup organizers (or others) lead and facilitate discussions at each local meetup. Then, crucially, we could gather summaries of feedback from each meetup and possibly even iterate this process. What I like most about this idea is that it engages a portion of the user community (at least potentially) different from "people who have the time and energy to keep up with GitHub," by taking the discussion to them. (I plan to write a future post about representation more broadly.)

Scaling with Decision Documents

Another way to reduce the load placed on the GitHub issue discussion would be to shift the focus to writing a "decision document" laying out the various points raised and presenting as fairly as possible both sides of the decision to be made. Then the discussion would serve primarily to suggest additions or changes to the decision document. This would have the important effect that someone who looked away for a week or two could catch up not by reading every message that arrived in the interim but instead by looking at what had changed in the much shorter document. We already do this informally for very large issues by trying to post summary comments occasionally; formalizing this in a separate document might help encourage people to start at the document instead of the discussion. The decision document could also be a new section in the proposal design document, but perhaps a separate document would be easier to point at. I filed issue 33791 to track this idea.

Summary

Overall, it seems clear we can do better at scaling and managing a single large discussion, but I think it is equally clear that process adjustments such as having many discussions or producing a decision document could make the final discussion easier and shorter, which would be a complementary, and possibly larg-

er, win. If I had to choose one aspect of discussions to focus on, I think I would focus on addressing the underlying social problem by taking steps to turn down the importance and heat of the final discussion, such as creating space for multiple discussions and introducing a decision document as a way to focus the energy of the discussion, instead of falling into the engineer's fallacy of "let's build a better discussion forum."

Next

Again, this is the fourth post in a series of posts thinking and brainstorming about the Go proposal process. Everything about these posts is very rough. The point of posting this series—thinking out loud instead of thinking quietly—is so that anyone who is interested can join the thinking.

I encourage feedback, whether in the form of comments on these posts, comments on the newly filed issues, mail to rsc@golang.org, or your own blog posts (please leave links in the comments). Thanks for taking the time to read these and think with me.

The next post will be about how the process for proposing large changes could be made smoother with some kind of official mechanism for experiments and prototypes.