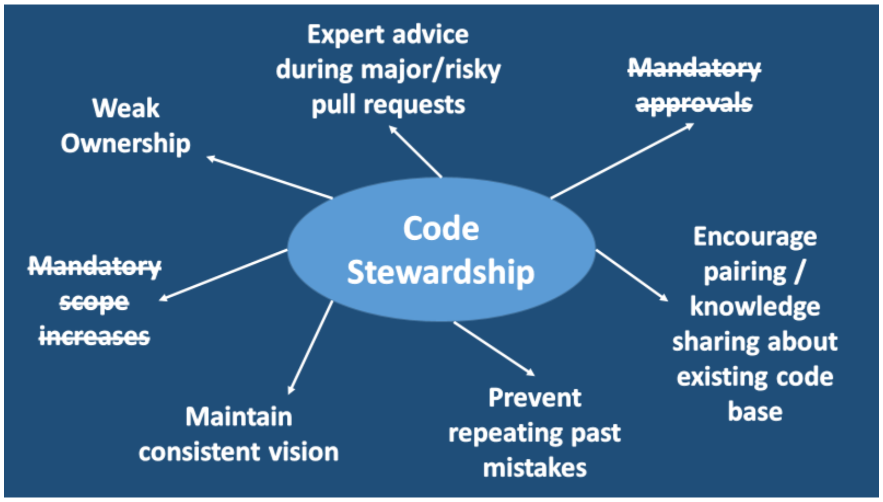
Code Stewardship – Summary and Reflection

What is Code Stewardship?

A code steward (or "owner") is someone who understands and cares enough about a particular piece of code that they're willing to provide advice and guidance to anyone else who is making changes to it. They may also initiate or lead cleanup/refactoring efforts for the code. Here's a mind map that describes some of the key characteristics and outcomes that usually result from implementing code stewardship at an organization:

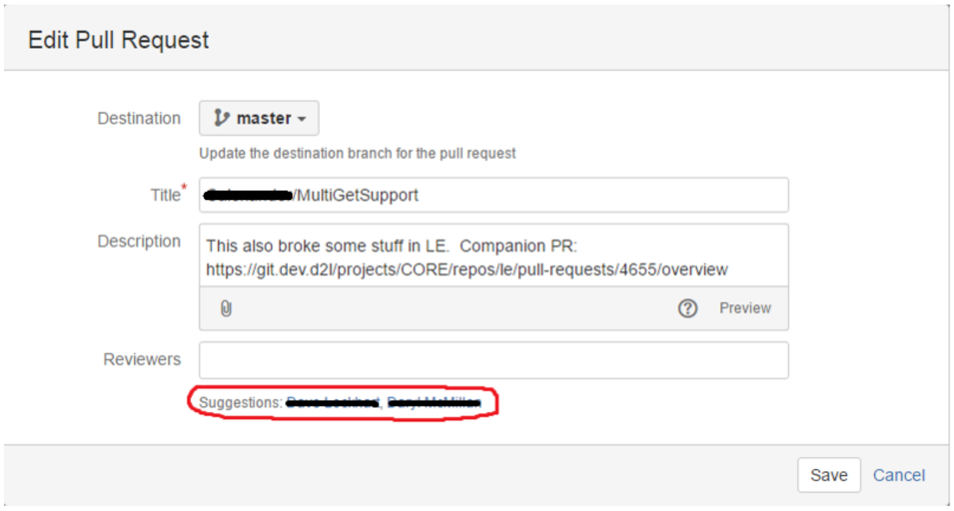

Code Stewardship
Expert advice during major/risky pull requests
~~Mandatory approvals~~
Encourage pairing / knowledge sharing about existing code base
Prevent repeating past mistakes
Maintain consistent vision
~~Mandatory scope increases~~
Weak Ownership

Some further clarifications and definitions according to Martin Fowler can be found here.

What history does Code Stewardship have at my company?

In 2016, the concept was introduced at my company and an open source plugin from Chromium was incorporated into our PR workflow in Bitbucket. A handful of people were recruited to sign up as owners for various areas within our largest and most complex repository.

When a PR is opened involving files that have owners identified, they are automatically suggested as code reviewers. It looks like this:



Edit Pull Request

Destination: master

Update the destination branch for the pull request

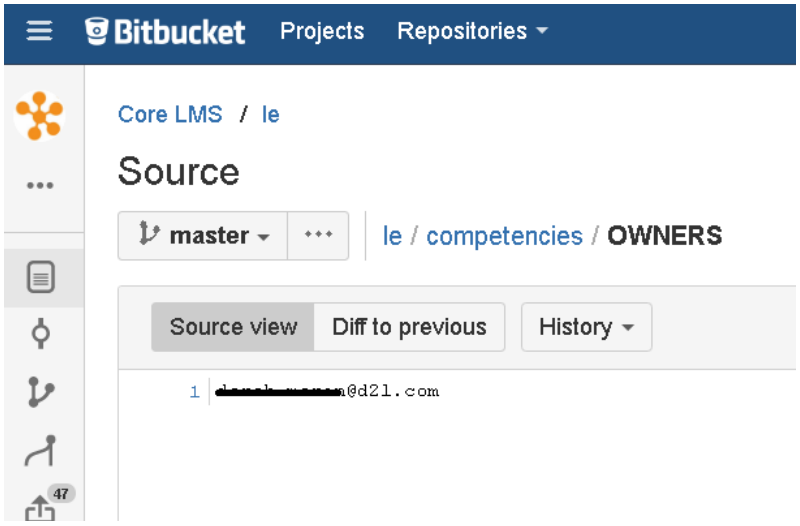
Title*: [redacted]/MultiGetSupport

Description: This also broke some stuff in LE. Companion PR: https://git.dev.d2l/projects/CORE/repos/le/pull-requests/4655/overview

Reviewers:

Suggestions: [redacted], [redacted]

To sign up as a steward, developers add their email address to an OWNERS file in the location of the code they'd like to own. The file looks like this:



```
[redacted]@d2l.com
```

Although the idea had a lot of promise, adoption of code stewardship didn't gain a ton of momentum beyond the initial investment of implementing the plugin and assigning a few high-level owners. In theory though, anyone could add themselves as an owner any time, now that the mechanism was in place.

What was I doing in 2017?

In 2017, I was looking for an initiative I could take on that would help to improve software quality at my company, with scope that extended across multiple teams. I really liked the concept of code stewardship, and felt it had a lot of potential beyond the level it was being used at the time. Best of all, the mechanics were already in place and all that was needed was additional influence and broader adoption (which are typically strength areas for me).

My plan for 2017 was to help expand awareness and adoption of code stewardship. By doing so, I also hoped that signing up as a steward would start to become a regular consideration for any developer who realizes they've become very familiar with a particular area of code.

My hypothesis was as follows: if we can increase the proportion of our code base that has stewards identified, the quality of changes being made to our product will improve. This will result from a higher frequency of well-informed, risk-focused discussions occurring during code reviews, which will be facilitated by the suggested reviewers plugin.

Why did I think this was a good idea?

There were three things that made me decide to invest in further adoption of code stewardship.

Potential ROI seemed good

I analyzed 60 escaped defect RCAs done in 2016. In my opinion, about half of them could have been prevented by better internal knowledge sharing. Each escaped defect is very expensive; the total cost includes taking a support call, routing it to the correct development team, diagnosing the problem, finding and implementing a fix, deploying the fix, and communicating the fix to clients and other stakeholders. On top of this, there is also usually a cost to our relationship with impacted clients.

Compared to the ongoing cost of escaped defects, the anticipated cost of expanding code stewardship adoption was relatively low. It would only require my own time spent on analysis and recruiting, plus (potentially) additional time spent by stewards doing code reviews. Therefore, it felt like a good investment overall.

Existing reasons for lack of adoption seemed resolvable
Many people were simply unaware of the concept of code stewardship, or didn't realize that the plugin was still active. This could easily be resolved through communication efforts.

Another reason that I suspected code stewardship hadn't fully caught on was that the same few names always appeared whenever there were suggested reviewers. This could be discouraging people from actually including them, as the change authors either wouldn't want to bother those same people all the time, or didn't have confidence that the suggested reviewers would actually have in-depth knowledge of what they're working on. This could be resolved by recruiting a more diverse set of stewards, especially in lower-level folders.

In terms of overall coverage for code stewards, there were many areas that just hadn't been explored for good ownership candidates yet. Some simple tools could be built to analyze data about recent changes, which would help to automatically identify developers who have been doing significant work in areas that don't yet have stewards.

Finally, it seemed that people who would otherwise buy in to the concept of code stewardship may hesitate to self-identify as an expert. There could be many reasons for this. Perhaps they were experiencing imposter syndrome or thought the bar was very high in order to sign up ("I don't know enough about this code", or "I have to know everything about this code before I sign up"). Maybe they simply couldn't recall all of the code they've worked on in the past. Maybe they just didn't get around to actually creating OWNERS files. In any case, I could actively recruit stewards, rather than wait for them to sign up, and show them data that proves they are qualified for the role. I could even offer to do the work of creating the OWNERS files.

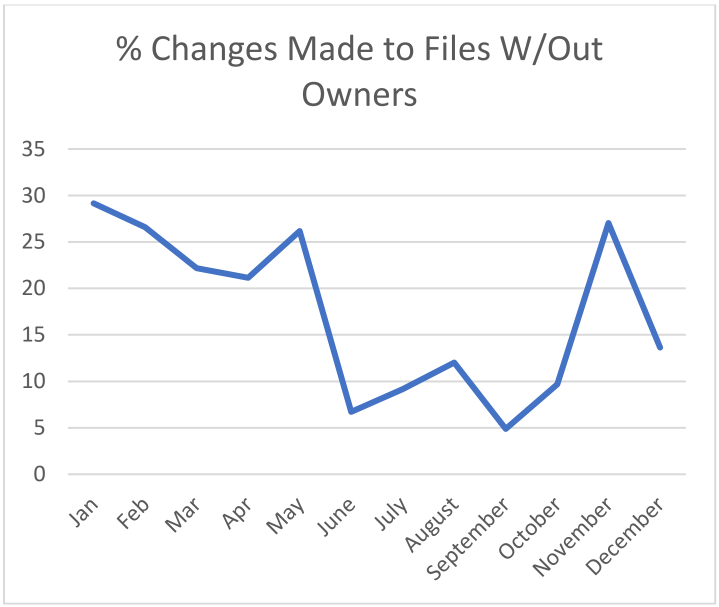
Progress seemed like it should be easily measurable
There are REST APIs available for Bitbucket that allow analysis of activity in our repositories (e.g. PRs opened, reviewers added, comments added, commits made, etc). I could use this data to help measure things like whether suggested reviewers are actually included on PRs, whether those reviewers provide input, and whether their input is acted on.

There are existing reports about our company's escaped defects, and RCA results are published regularly; I could use these to observe trends in overall escaped defect quantities, and in the reasons for escaped defects. I could also analyze the location of OWNERS files within the repositories to help understand where stewards exist, and how close they are to any particular code file. Finally, doing a code search for new OWNERS files added in our repositories could indicate whether developers are starting to sign up for stewardship on their own, rather than waiting to be asked (which is needed in order for the concept to become self-sustaining in the long term).

Why I'm still not sure if I actually made an impact
It was true that there were lots of things about this work that were easily measurable, but for each data point I was tracking I could always think of reasons why it could be indicating positive progress, and other reasons why it could also be meaningless.

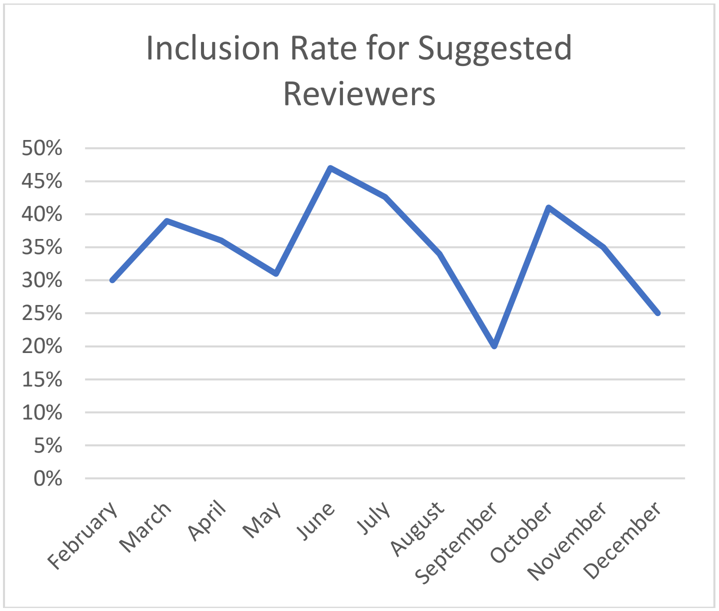
Here is some of the data I collected. Note: for the purpose of my initiative, I focused on only one of our repositories (the largest and most complex one).

% of changes made to files without owners
If this number is decreasing, it should mean that changes being made are safer overall (or at least have the opportunity to be safer, if suggested reviewers are actually included). On the other hand, as a particular area of code became active, I would recruit new owners in that space and so the % of changes being made to files without owners would go down again for a while, until a different area became active instead and the overall % would jump back up.



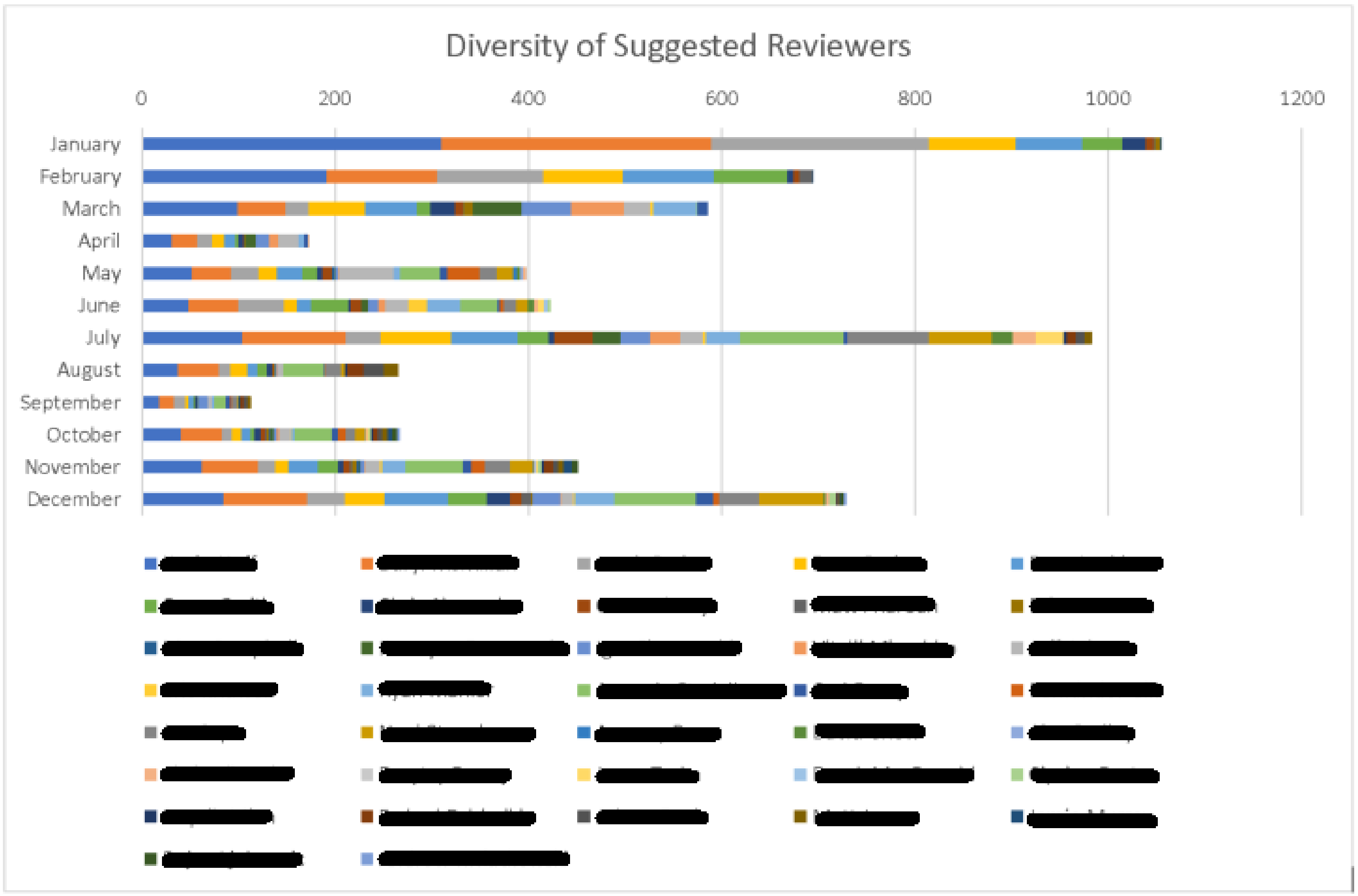
Inclusion rate for suggested reviewers

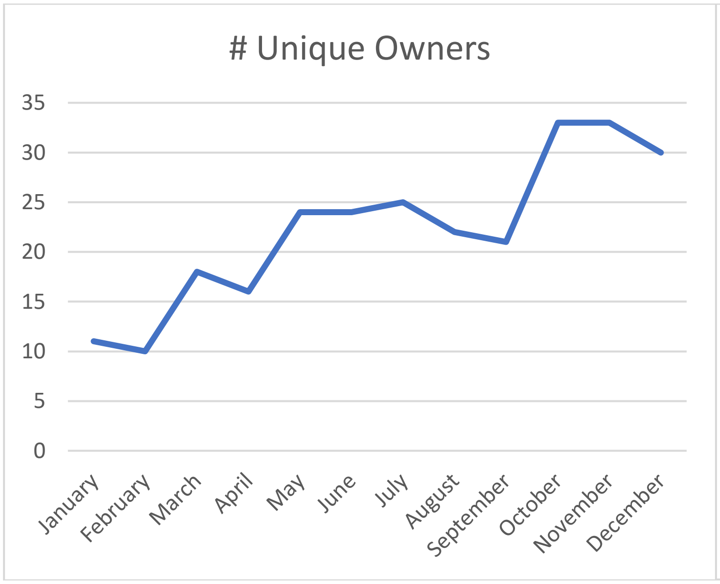
If this number is increasing, it should mean that safer changes are being made due to the inclusion of knowledgeable reviewers. It could also mean that the quality/credibility of suggested reviewers is improving, if people are more willing to include them. On the other hand, some changes are low risk and may not benefit much from bringing in an expert, so it may not make sense to strive for a very high rate anyway. Also, changes vary a lot in size, and so developers who tend to make lots of small changes have a much greater impact on the overall rate.

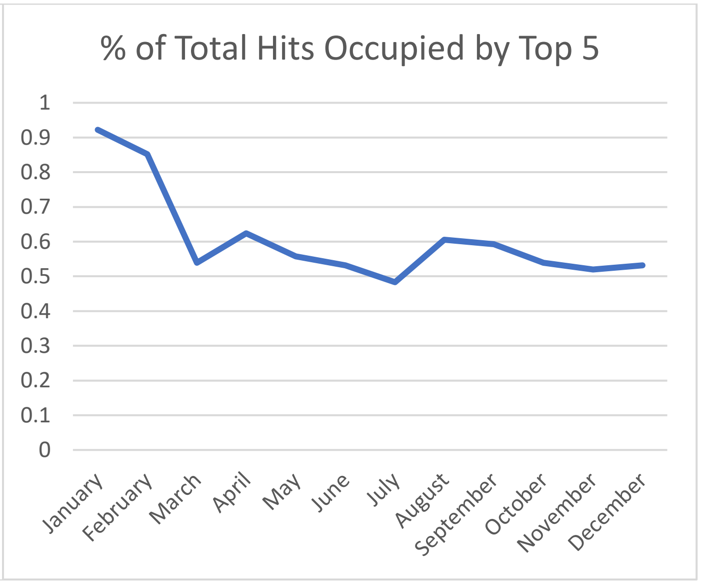


Diversity of suggested reviewers

If there is greater variety in the overall pool of suggested reviewers, change authors may be more likely to include them as it will not feel like the same few people are being overloaded with review requests. Whether that assumption is true or not, this is one case where the data definitely shows an improvement.

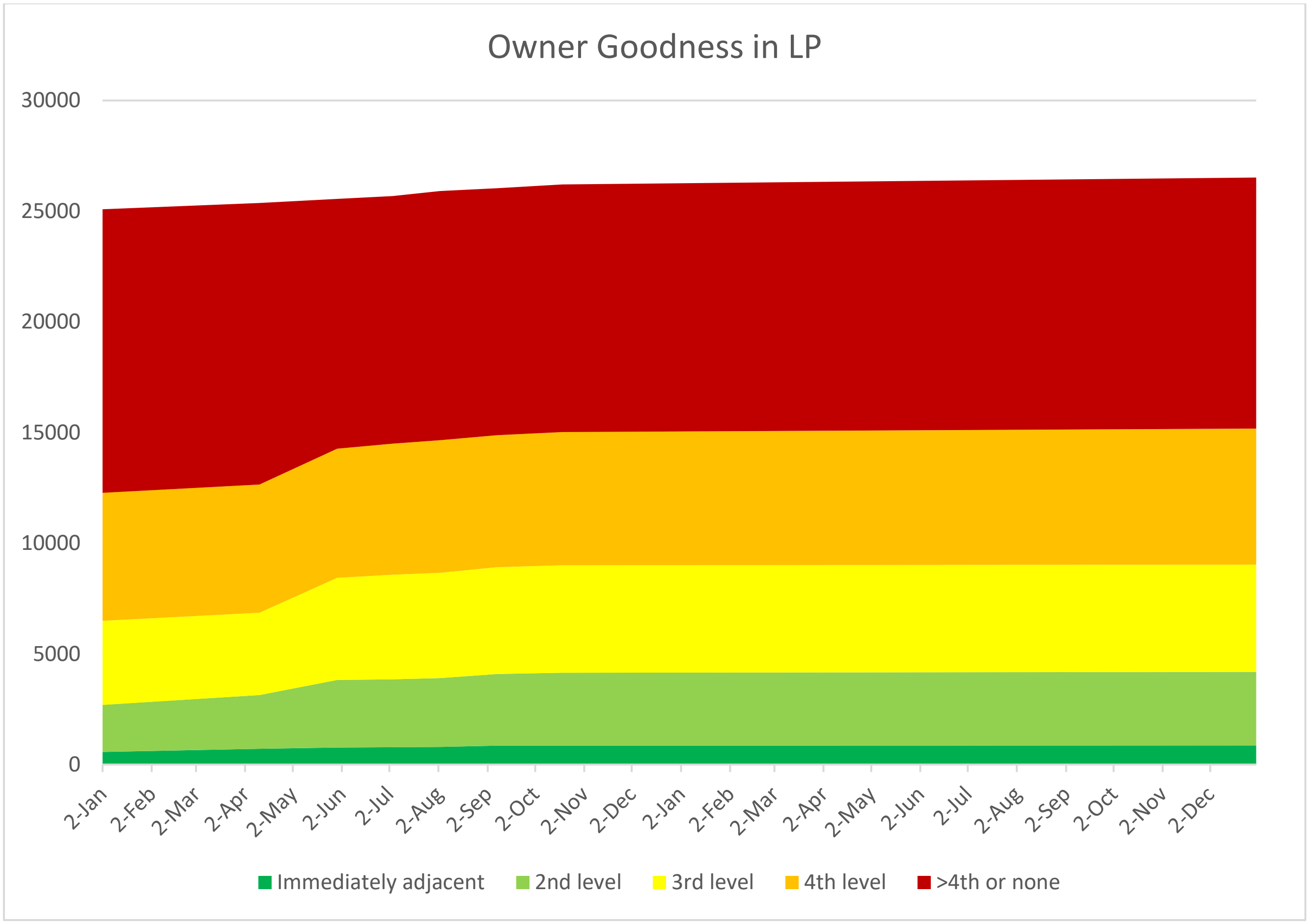






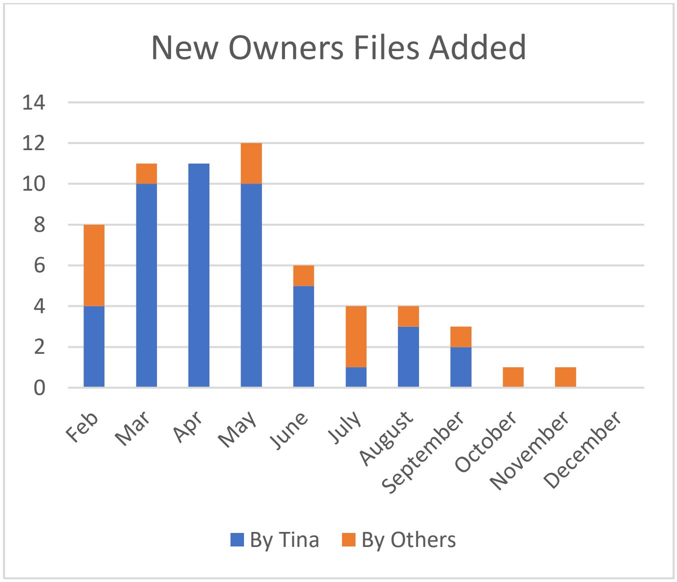
<u>Distance from each file to closest steward</u>

Code stewardship is hierarchical; if no OWNERS files exist in the same location as the files being modified, the plugin will check folders higher up in the repository hierarchy until a steward is found. In many cases, the further away a steward is from a particular file, the less detailed knowledge they are likely to have about it. Therefore, it may be better to have stewards with very detailed knowledge placed further down in the hierarchy, rather than stewards that are placed close to the top of a repository. On the other hand, maybe stewards that are too "localized" do not have bigger picture knowledge that could be very useful as well. In any case, although lots of new stewards were added throughout the year, it was not enough to make a significant dent in the overall number of files covered by stewards in the repository I chose. It's a large one, and continues to increase in size.



New owners files added by people other than me

In order for the concept of code stewardship to survive long term, it eventually needs to become self-sustaining (i.e., adding oneself as a steward in areas where in-depth knowledge is gained becomes part of the development culture). If self-additions start to increase in frequency, this would be a sign that the concept is gaining long term traction. It's unclear where the tipping point would be that would mean the practice has been fully accepted.



Escaped defects trends

The overall quantity of escaped defects actually increased during 2017 (unfortunately, the report I was formerly using is no longer available to include in this article). However, it is extremely difficult to tie any code stewardship-related activities directly to escaped defect trends, as there are so many factors that influence and contribute to these events. Also, RCAs were not published in as much detail in 2017 as they were the previous year, so it is difficult to say whether defects that could have been prevented due to improved internal knowledge sharing were on the rise or not.

What I definitely gained from the experience

While I'm not confident that my work on code stewardship had any material impact on software quality at my company, I did achieve some things that I believe were valuable.

First, this initiative gave me a great opportunity to write some code again (something I hadn't done much of for quite a while). I built several tools that helped me analyze activities and files in our Bitbucket repositories, using the Bitbucket REST APIs. The Chromium plugin sends emails to a distribution list in cases where no suggested reviewers were available for a pull request, and I also built tools that analyzed these emails. Finally, when proposing new code stewards, I submitted the PRs to add the relevant OWNERS files myself – both to save time and hassle for the new owners, and also to give myself experience with committing code to our production repositories.

After analyzing the data and coming up with code stewardship candidates, I also had a good excuse to introduce myself to lots of new developers I hadn't had a chance to work with before. I learned about what areas they were knowledgeable in, and I often chatted with them about their views on working safely with legacy code. I also followed up with people that I noticed were making frequent use of code stewards as code reviewers, and asked them about their experiences with doing so. All of these conversations also provided me with good anecdotal data points about overall sentiment towards the code stewardship concept.

Finally, coming back to my original hypothesis – well-informed, risk-focused discussions may be occurring *somewhat* more frequently as a result of my efforts to sign up new code stewards. Whether or not this actually translates to fewer escaped defects, better quality, or any other measurable benefit is definitely still up for debate. That being said, I know I learned a lot from doing this work. I have new knowledge about our code, new skills that help me work more effectively with it, and new relationships I likely would not have built otherwise. As an individual, I am better positioned to produce better quality work in the future. And that's probably still a pretty good outcome.

Where to go from here

I hope that people will continue to leverage the code steward concept where it makes sense. But even if no more new owners ever get added, the ones that are already there will still be useful in some cases in the future.

In July 2017, GitHub introduced native support for code owners. So, we could start leveraging this capability in our GitHub repositories any time as well.

Although I believe I could do more useful work related to influencing code stewardship adoption, I'm going to focus on other things in 2018 (such as getting better in my new role as a development manager). I'm always happy to chat about anything related to this topic, though – including deeper dives on any of the data I collected, what it might mean, or what some useful next steps might be for anyone else who wanted to work on expanding this idea further.