Investing in Code Reviews for Better Research Software

Date: September 7, 2022

Presented by: Dominik Krzemiński (University of Cambridge), Thibault Lestang (Imperial College London) and Valerio Maggio (Anaconda)

(The slides are available under “Materials from the Webinar” in the above link.)

Q. Do people have recommendations for / experience of code review practice when working in small research groups with few technical peers? e.g. setting up groups that "cross borders" between research teams? What are the challenges?

A. Partially covered in the slides:
https://tlestang.github.io/HPCBC-code-review-12102022/slides.html#/specify-the-feedback-you-are-after/1
It works best when there is a mixture of people from various backgrounds, but the discussion is monitored by more experienced software engineers.

Q. Quality is linked to economics. For commercial software, you want people to buy it. so you work on documentation, ease of use and customer support and so on. For research software, you often do not care if anybody else uses your code. you only care about publication. Users outside of your research group are often of secondary importance. You may answer a few questions here and there if you are nice. but you are too busy to provide customer support when there is no reward. Given the economy of things, what are some extra factors that you can inject into research software to boost accountability?

A. The economy aspect is important, but the only one. We believe that researchers should change a mindset and notice the benefits they give without any external incentive. Some example include:

- Better codebase after students/postdocs leaving the lab;
- Increased reproducibility.

Q. How do you break “the routine” and ask for such (live) code reviews in a team that is already working? How do you make it look less like overhead and more like an opportunity?

A. The important aspect is engagement in the project. If a reviewer feels that the code review is relevant (also for them) that definitely increases the focus.

Chat transcript (edited and anonymized, California times):

10:02:37 From participant-1 to Osni Marques(Direct Message):
I do agree that code review is critical for better software but what I discovered recently working with two individuals, one with a PhD from a good university and working at a national lab, publishing specific one time CUDA code, and another who graduated from a prestigious Master’s program, also working at a national lab and now at a prestigious university, when I

asked them to write "general" code that could work on many machines and environments they were clueless. What I believe is also important in addition to code review is program design in a general way so that it does not have to be rewritten over and over.

10:22:52 From participant-2 to Everyone:
quality is linked to economics. for commercial software, you want people to buy it. so you work on documentation, ease of use and customer support and so on. for research software, you often do not care if anybody else uses your code. you only care about publication. Users outside of your research group are often of secondary importance. you may answer a few questions here and there if you are nice. but you are too busy to provide customer support when there is no reward. Given the economy of things, what are some extra factors that you can inject into research software to boost accountability?

10:30:17 From participant-3 to Everyone:
@participant-2: I would argue that making your research software easier to use most commonly will lead to collaborations with others, and potentially more publications.

10:32:31 From participant-2 to Everyone:
@participant-3. Yes, I agree. I have worked with researchers who are very willing to collaborate with random people. but I think the motivation comes from wanting to promote their codes--especially if you are part of a very influential big research group.

10:40:17 From participant-2 to Everyone:
I have seen quite a few times that researchers put a patch out there even more than they compile it. the codes don't even compile and we have no way to do anything else. There are many responsible researchers and some others are not as many. The community often does not punish bad practices nor reward good practices. so things are the way they are sometimes.

10:40:54 From participant-4 to Everyone:
Looking at a local public university, even for computer science majors, good software practices such as code reviews, continuous integration, etc., are not taught. If it is not required for CS majors to learn this, how can we possibly expect other disciplines that code to actually use good software practices. Perhaps other universities actually teach this, but not here.

10:43:25 From participant-5 to Everyone:
@participant-4, I think what you are saying was true in the past, but in my experience Universities are now starting to cover code review, continuous integration, etc.

10:47:11 From participant-4 to Everyone:
@participant-5. Good to know. Now if we can just get this required for other disciplines that write code.

10:50:35 From participant-4 to Everyone:
Even a really great set of experienced software developers really struggle to agree on a "nice" set of required software practices. Lots of incredibly great opinions does not necessarily make it easier.

10:52:48 From participant-3 to Everyone:

@participant-4: In CSE courses I am involved in (TU Munich), we ask students to do code reviews of others' codes, and we show a bit of CI. Every year, we add more and more of these components.

10:57:51 From participant-5 to Everyone:
@participant-4 I have integrated git, containerization, and team based projects in all my courses. As participant-3 said the complexity increase as the progress towards their senior year

10:58:29 From participant-5 to Everyone:
Great talk! Thank you! Something else to consider is to apply the code review ideas to other software artifacts. I know people don't tend to love documentation, but the quality of documentation is very important. Reviews of documentation should be done (in my opinion) with the same rigor as reviews of code. :-)
.
11:07:17 From Dominik Krzeminski to Everyone:
Very good point @participant-5, I agree 100%