Discovering and Addressing Social Challenges in the Evolution of Scientific Software Projects
(the slides are available under “Presentation Materials” in the above URL)
Date: September 11, 2019
Presented by: Rene Gassmoeller (UC Davis)

Q. For developer meetings, is there a recommended format for what topics to be discussed? Do you have small, frequent meetings to focus on a particular topic (e.g. infrastructure) to make sure the discussion doesn’t derail to other topics? Or do you have a large ‘conferences’ with different break out sessions?

A. In my opinion large meetings are better for fostering the community, but smaller meetings (e.g. only the core maintainers) are better for critical infrastructure changes.

Q. As a “hero”, how to best “step back” and encourage more community?

A. First of all make sure people know that you want to grow into a community code. Then make sure they get recognition for contributing (e.g. listed as contributors). As a last step (when you have some members in your community) you need to hand over some responsibility as well to get them more strongly involved.

C. I see a conflict in management philosophy. On one hand, you want to a community where many people contribute ideas. But when there are too many ideas, you cannot move forward. Democracy is very inefficient. Sometimes you need a dictator to give a single cohesive direction in order to move forward. In Apple, they used to have Steve Jobs. In a community, there are competitions. Some people want to maintain their leadership. Others who contribute ideas that are not in line with the status quo may get shot down.

A. Certainly sometimes a decision has to be made. However the case where the community does not come to a conclusion is somewhat rare and there are successful projects (think Debian) who work without a benevolent dictator. It depends on the environment and the personality of the leaders whether a benevolent dictator is helpful. It also depends on the application how often a hard decision has to be made by a small group of people. I do not think there is a general consensus about which approach works better.

Q. In your experience, what proportion of scientists only openly publish finished code (after the paper is submitted) vs develop code ‘in the open’ (cf. ‘open science’ principles). Is there a way of encouraging more openness ?

A. The proportion of scientists who develop in the open is hard to guess, it also depends on the definition of 'open'. But in my opinion ‘Many people are too afraid of being scooped’ - the benefit of open development is that others may cite you even whilst you are solving your own problems. While it is acceptable to keep the code back while you are still unsure if it works at all, once it works I think the benefit of releasing the code outweigh the risks of doing so.

Q. Do you have any references that you could recommend for learning and practicing different governance methods? Are there non-software focused resources that you found relevant or helpful?

A. An extremely helpful book is “Producing Open Source Software” by Karl Fogel that also describes a few different governance models for software projects. I missed the part about non-software resources during the talk, but while I am sure there is material about this, I can't immediately think of one particularly helpful resource.

Q. If not at a meeting, when or how should work on building or refactoring infrastructure be targeted?

A. I assume this question refers to my opinion to not overthrow critical infrastructure during community meetings. It is for example possible to hold a smaller, very focused meeting, or a virtual meeting (set aside a week for the involved people and work on it) that deals with only this aspect to make sure it generates as little conflicts as possible.