The Challenges of Transitioning to Peer Production

Johanna Cohoon
Caifan Du
James Howison
School of Information at UT Austin

Scientific software is an important tool and research output that is used to discover, share, and reproduce knowledge. However, it requires maintenance and continued development to remain scientifically useful over time. One approach to sustaining useful scientific software is to establish an open-source development community, also called a peer production community, around the software so that anyone can contribute to its maintenance. Funding agencies like NSF have shown favor toward this approach, establishing funding initiatives like the Software Infrastructure for Sustained Innovation (SI2) program which sought to establish "sustainable software communities" (*Software Infrastructure for Sustained Innovation (SI2: SSE & SSI)*, 2016). To understand how these sustainable peer production communities (PPCs) can be developed within the timespan of a grant, we empirically studied 113 SI2 funded software projects. We reported early results from this study at the 2019 Workshop for Sustainable Scientific Software. The present report includes substantially more data; the findings are nevertheless preliminary as data collection recently concluded.

We reviewed current and archived websites associated with the funded software projects, recording information on web content and features that the literature suggested might impact the success of a PPC (*e.g.,* the presence of a public code repository). To enrich our analysis, we interviewed project leaders and contributors from a subsample of these projects, seeking detailed explanations of work episodes (*e.g.,* when a person outside the core team contributed code) and descriptions of how the project organization has changed over time.

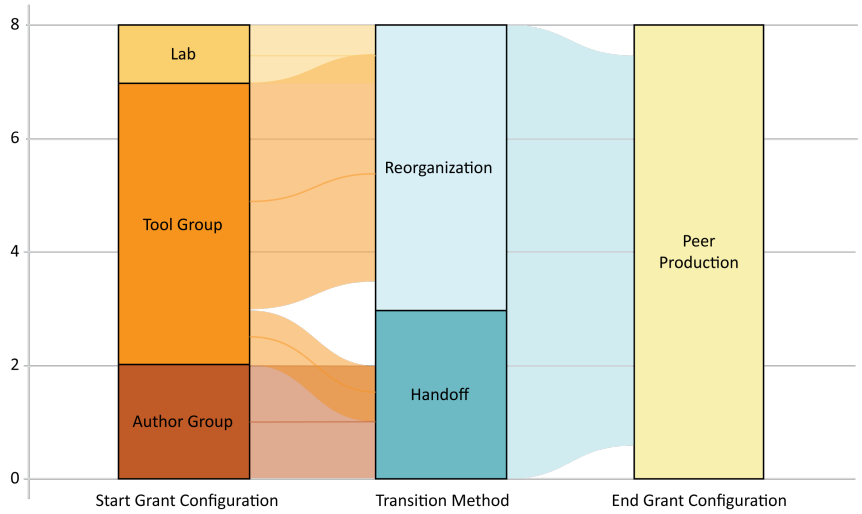
Through our qualitative study, we found that scientific software projects might transition to peer production via several routes but that they rarely do so during the grant period. We have mapped project transitions across two directions. First, as seemingly envisioned by the SI2 program, software projects might transform the development approach to peer production. This might require, for example, establishing means for outside contributions via distributed source management. We refer to this manner of changing the method of work and organizing as a *configurational change*. Second, a scientific software project might change the makeup of its organization, i.e. undergo *organizational change*. In this route, the development group responsible for the software turns over such that the original developers are no longer the primary maintainers of the project. Software projects may go through configurational change, organizational change, or both during a grant period. We term the various combinations of these routes to change *reorganization*, *hand off*, and *migration*; any might lead to peer production.

Our findings showed that a transition of any kind during the SI2 grant period was unlikely—89 of 113 software projects made no transition. The most common form of transition was a *handoff* where the software project underwent organizational change and configurational change and was thusly developed by new people in a new manner. We saw 13 projects undergo a handoff during the grant period. *Reorganization*, where the development

group keeps its core members but changes its configuration, was the second most common method of transition in our sample; nine projects took this route. Finally, two projects *migrated* their software from one organization to another of the same configuration.

We saw that scientific software was produced in organizations with different configurations. Our findings suggest that these configurations affect projects' choices and challenges when pursuing sustainability. In this report, we focus on those software projects that transitioned to peer production, shown in Figure 1. These transitions originated in three configurations: labs, tool groups, and author groups. We defined a lab as a named group with localized and closed membership, diverse research/software interests, and a web presence that usually centered around the PI. Tool groups differ from labs in that they focus on a single research/software project and they may be made up of collaborating but non-collocated members. Finally, author groups were defined by their informal organization, lack of web presence, and lack of invitation to contribute (*i.e.,* closed membership). Below, we discuss stories of transitions made to peer production, focusing on challenges experienced by each of these starting configurations.

Figure 1. Types and Counts of Transitions to Peer Production Communities



Note. Eight software projects in our sample transitioned to peer production during the grant period. None of them demonstrated migration as a transition method.

Transitions by Author Groups

One quarter of the transitions to peer production in our sample were initiated by author groups (2/8 transitions to PPC). As informal and transient collaborations, author groups need not to establish the same processes and infrastructure that is required of a vibrant PPC (Katz et al., 2016). However, if they will not reorganize to become a PPC but they still seek sustainability via peer production, they must hand off their software work to an already established PPC. This requires that the author group members learn and adopt the infrastructure and processes of the PPC they will contribute to.

In our interviews, we found that this can be an arduous process and sometimes requires the PPC to bend (perhaps with resistance) to the practices of the author group. One PPC

maintainer that received a code handoff from an author group expressed frustration when describing the process of merging a patch from an outsider: "…I found it difficult to mentor him. He has his own style of writing software and he's not terribly interested in adapting to the rest of the project…. I probably spent multiple days just on the review….When you get 1,500 lines of code, you're like, 'Oh my god, how am I gonna do this?'…I mean, if it's 100 lines of code from one of my co-authors, I know there's not much for me to do" (interview 21b).

Transitions by Labs

We saw only one lab transition their software to be maintained by a PPC (1/8 transitions to PPC); they accomplished this through reorganization. Prior work and our current interviews showed that researchers seeking to advance their scientific careers perceive open-source software work as an impediment toward their advancement (Howison & Herbsleb, 2013). Thus, labs, which are concerned with demonstrating and effecting the PI's long-term impact on their field, might be reticent to reorganize themselves as PPCs. One interviewee described a prior experience: "We would sometimes get requests for the code like, 'Hey, can I use your model to do XYZ?' At the time the director of the lab said, 'No,' said, 'Look, the model is the lab. You wouldn't just let anybody off the street to go into your lab that you worked so hard to set up, and craft, and manage, and just give them a competitive advantage.' So during those years, we were not sharing the code except among the kind of alumni of the group" (interview 17a).

Transitions by Tool Groups

We saw that a majority of the transitions to peer production were accomplished by tool groups (5/8 transitions to PPC). We defined tool groups as a configuration similar to peer production communities—they both organize around software, they both have strong identities and online presences, but they differ in the degree to which they welcome outside contributors. However, inviting outside contributors is more complicated than simply adding a sentence to the project website; significant social challenges need to be overcome as well. A sentiment expressed in more than one transition from tool groups to peer production was that more contributors and collaboration meant less control. One interviewee described an "anti-collaboration" philosophy present in their early days where the team members expressed concerns about "having a whole bunch of people jumping in the project, pulling the project one way or another. 'We are gonna lose control of the code, lose control of the architecture!'" (interview 11a).

However, even after inviting outside contributions and ceding some control to those newcomers, these newly minted PPCs still maintained responsibility for soliciting funding. One software project left behind their parent organization and reorganized into peer after inviting outside contributors and establishing their own web presence. As described by one interviewee, this federated community was made up of package developers with "tremendous authority to dictate" their paths. "Yet, at the same time, [our project] as a whole, as an aggregate, is what our funding sponsors pay attention to. They don't care that there's a package called [so and so] in it, they care that it's [our project]" (interview 4a). Through this transition process, the project became a PPC and a parent project to the diversity of packages that they manage. While this required them to risk losing authority over the code by inviting more developers to the table, they maintained control over the project funding and the related

opportunities for long-term planning. Thus, a transition from tool group to peer production involves much more work than simply extending an invitation—it endows new roles and responsibilities on the developers undertaking the transition.

Continuing Peer Production

Most often, software projects that concluded their grant period as a PPC also began the grant period that way (13/21 projects ending in peer production). Some of these non-transition cases described their experiences with establishing a PPC in interviews with us.

Continuing the above-mentioned concerns about control, tracking impact seemed to be a special challenge for PPCs because of the control acquiesced by the core developers to a community that might choose to fork the code. One interviewee reported, "Back when we could track this more easily, when people would actually download software, we had at one point 10,000 unique user IDs worldwide. Now we have no clue. Github doesn't allow us to really track that in the same way…We know of teams that there's one person who's in charge of that and everybody works off of theirs. We have no idea of the usage base anymore" (interview 4a). One PPC attempted to prevent forking by "aggressively" developing new features, putting out four major releases per year thus making it difficult for forks to keep up (interview 20187[1]).

Some decisions about PPCs' code were made based on the input of trusted users or colleagues. This was facilitated by collocation—one interviewee said, "[A colleague] was in this office directly opposite ours. So a lot of times when [my collaborator and I] were discussing things on a whiteboard, [this guy] would come in. He would ask 'Hey, what…stuff are you guys working on right now?' He would often have suggestions and better ideas than [we] would come up with" (interview 19a). Our interviews showed us that close personal relationships like the one described here were crucial to PPCs' success. Furthermore, those cases demonstrate that PPCs rely on their communities not just for code contributions or even documentation updates, but also for the productive attention of users.

Other stories shared during interviews indicated that PPCs made decisions about their architecture based on the labor that was available to them. For instance, one project enabled plugins so that students could become familiar with the code more easily: "When we made it a monolithic, single, executable, that's when we built in the plugin concept…That lowered the barrier for a lot of students and a lot of newcomers…And as they got used to their own plugins, they would learn more…and eventually be able to contribute back to the infrastructure" (interview 3a). This move was met with limited success, however, as students, rather than other newcomers, continued to be key sources of contributions.

Our emerging findings indicate that scientific funding can help establish peer production, but that this happens only occasionally during a given grant period. Software may be developed within different organizational forms and these various forms face different challenges to successfully transitioning to peer production. Our current work is refining the results from our interviews to bring life to these challenges.

[1] This scientific software project was not funded by SI2 but was part of a theoretical sample of peer production projects we studied early in this research.

References

Howison, J., & Herbsleb, J. D. (2013). Incentives and integration in scientific software production. *Proceedings of the 2013 Conference on Computer Supported Cooperative Work - CSCW '13*, 459. https://doi.org/10.1145/2441776.2441828

Katz, D. S., Choi, S.-C. T., Wilkins-Diehr, N., Hong, N. C., Venters, C. C., Howison, J., Seinstra, F., Jones, M., Cranston, K., Clune, T. L., de Val-Borro, M., & Littauer, R. (2016). Report on the Second Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2). *Journal of Open Research Software*, *4*, e7. https://doi.org/10.5334/jors.85

Software Infrastructure for Sustained Innovation (SI2: SSE & SSI). (2016). National Science Foundation. https://www.nsf.gov/pubs/2016/nsf16532/nsf16532.htm