Open Source Community Governance the Apache Way

Isabel Drost-Fromm and Rob Tompkins, The Apache Software Foundation

An open source project without the people is a dead project—or at least one that is fairly deep asleep. While all successful open source projects understand that they need to build a community around their project, the exact options for doing so differ.

A considerable variety of open source projects find usage across the various computing appliances seen at use in our world today. By "open source," we refer to a software project that is freely distributed, and the code base is freely hosted.[1] For such a project to exist and gain success over time, a number of people must necessarily contribute. Indeed, some projects of note have solitary contributors and owners. However, in our experience, we have seen that well-governed communities of contributors yield greater success over time. Furthermore, healthy projects rely upon standard principles that encourage conversation and community building.

Open source projects come in all shapes and sizes, from the tiny tools maintained by a single developer to the multinational projects that span several continents, cultures, time zones, and businesses. As soon as more than one person is involved, there are a couple of questions.

- Which name and address should be used when registering accounts for project management, social media sites, and so on?
- How will we pay for the infrastructure bills, both in terms of where the money comes from and which account to use for paying those bills?
- What kind of work do we need help with, and how do we motivate others to provide that help?

Digital Object Identifier 10.1109/MC.2021.3058023
Date of current version: 9 April 2021

0018-9162/21©2021IEEE

EDITOR **DIRK RIEHLE**
Friedrich Alexander-University of Erlangen Nürnberg;
dirk.riehle@fau.de

FROM THE EDITOR

Welcome back to the "Open Source Expanded" column and the current theme of open source communities! After last column's summary of the history of open source communities, we will now look at one of the earliest and most prestigious open source foundations, The Apache Software Foundation (ASF). I'm glad I could convince Isabel Drost-Fromm, a long-time member and mentor of the ASF, and her colleagues to explain to us how ASF-style governance helps open source projects succeed. Next up will be governance at the Eclipse Foundation. Happy hacking, everyone, and please stay healthy! — *Dirk Riehle*

- Who owns the project trademark? Will that be a separate entity?

As an example, consider the creation of the Apache Mahout project. Apache Mahout was born out of an email thread in the Apache Lucene Project's mailing list. A simple conversation about the potential of a text mining project led to several people collaborating to start Mahout's code base. We used systems hosted on the private webserver of one of the project founders. As the idea grew, we needed to decide where to chat, which mailing lists to use, where to store source code, and which issue tracking service to use. In the end, we moved the collaboration to The Apache Software Foundation (ASF). The goal was to make it independent from individuals in the community and instead set it up for a future backed by a diverse community.

Very basic infrastructure decisions like this taken very early in the project lifecycle decide what the open source project will look like for years to come. The most basic question to ask early on is about balancing control with project growth and longevity. How important is it to the initial project founders to retain control over the course of the project? How much control are project founders willing to delegate in return for allowing the project to grow faster and become independent of the individual members?

GROWING BY DELEGATING

Initially, most projects are driven by one or a few dedicated individuals—there's one project maintainer driving development, issue triage, and customer support. As a project grows, this individual turns into a bottleneck unless a healthy community can be grown. For example, one developer said, "So you can either try to drink from the firehose and inevitably be bitched about because you're holding something up or not giving something the attention it deserves, or you can try to make sure that you can let others help you. And you'd better select the 'let other people help you,' because otherwise you will burn out. It's not a matter of 'if,' but of 'when'."[2]

As one example, look at Linux. For a long time, the project was known for following the benevolent dictatorship pattern. It relied on one single individual to take over the stewardship of the project. This changed only a few years ago, where the maintainership of major modules—and, at some point in time, even the kernel as a whole—is backed by multiple people.[3]

The ASF takes a different approach. Apache projects are expected to be run by a community of individuals in a vendor-neutral way. As a result, projects are a neutral ground where even competitors can work together to create something larger than what any individual participant could create. What that implies, though, is that no single participant has full control over the project direction.

Yet another option is open source projects owned by a single commercial entity. Often the goal there is to retain full control over the project direction. These projects were described in more detail earlier in this series as single-vendor open source projects.[4]

At the end of the day, the motivation for creating the project drives which governance model is ultimately adopted. One good overview of open source project archetypes was published by the Mozilla Foundation.[5]

> Open source projects come in all shapes and sizes, from the tiny tools maintained by a single developer to the multinational projects that span several continents, cultures, time zones, and businesses.

MOTIVATING COMMUNITY MEMBERS TO CONTRIBUTE

The goal of every open source project is to pull people in to help the project along. One of the most promising vectors to achieve that is to make use of intrinsic motivators, including the following:

- *Autonomy*: The more people who feel like they are able to make their own decisions independently of others, the more likely they are to participate.
- *Mastery*: The more participants feel like they can improve their own skills, the more likely they are to put more energy into a project.

- *Purpose*: To participate meaningfully in a project, people need to understand the purpose of the project but also the impact that their contribution has on the project.

With the commercialization of several open source projects, some of the motivation to contribute has changed by now, although the superiority of intrinsic to extrinsic motivation is still valid.

Work transparently: A precondition for allowing contributions

The goal of open source projects is to turn users into active contributors. As a result, more than just the source code needs to be available. Instead, it should be easy to follow the entire development process.

- Design documents should be available.
- Communication on current changes to the code base needs to happen in public.
- Roadmap planning needs to be available publicly.

COMMUNICATE EXPLICITLY WHERE HELP IS NEEDED

Any new contributor coming to a project will need information on the general project mission—on what is under development. Figuring out what the project really needs help with can be tricky, though. At Apache Mahout, for a long time, contributors thought that the best way to get involved would be to add more machine learning algorithm implementations. Instead, the project was looking for better documentation, help with answering questions on the mailing list, better test coverage, and performance optimizations of existing implementations. Making those areas explicitly documented increased the collaboration and drove contributions to areas where the project really needed help.[8]

> The most basic question to ask early on is about balancing control with project growth and longevity.

Projects at the ASF have a very simple rule of thumb for deciding what needs to happen visibly to everyone. Every decision that is vital to the project needs to be made in public—for everyone to see and for everyone to participate.

As simple as that sounds, for new projects coming to the ASF this is often one of the hardest things to learn. That's why, early on, projects at the ASF undergo an incubation period where one of the main goals is to move all of the communication to public communication channels. (see Delacretaz[6] and Curcuru[7] for a description on how this works using the example of how the ASF uses mailing lists as a communication channel.) Often, the result of this process is that people other than the original project owners become active and part of the development team.

The level of transparency described directly supports contributors gaining autonomy. To create a contribution, there is no need to wait for others to take the time and explain things such as the project architecture or direction. Instead, all of the information is available on a self-serve basis.

Making extensive documentation about the project transparently available also means that contributors can improve their own skills, thus directly feeding into mastery as an intrinsic motivator.

Making help requests explicit also means that contributors understand the purpose of their contributions. They better understand how what they do fits into the bigger picture of the entire project.

SLOWING DOWN TO MOVE FASTER

Many projects start with the intention of answering every question immediately, fixing every bug very quickly, and moving fast.

If the goal is to make contribution possible for more people, though, it can help to slow down. Instead of answering every question yourself as a maintainer, leaving questions open means that other community members can step up and help with user support. Leaving trivial bugs open and marking them as such helps new contributors as they then have trivial changes they can use to explore how to check out the code, make the change, build it, run tests, and submit changes.

Leaving discussions open for a few days helps with integrating people in many different time zones; often, moving fast means that decisions are made while interested people are well asleep. As a result, for major decisions at Apache, we have a rule to leave discussions open for at least 72 h before a final decision is made. That way, even if there's a public holiday for some of the contributors, there's still a good chance they can weigh in another day. Technologically, this means moving from synchronous communication like video conferences and chat systems like slack to asynchronous communication like archived, searchable mailing lists.

Translated to intrinsic motivation, one aspect of slowing communication down means that contributors feel more autonomous in making contributions. Instead of being dependent on the original project authors when it comes to communication schedules, contributors can participate according to their own schedule.

SCRATCH YOUR OWN ITCH MOTIVATION

The group of people with the highest motivation to contribute are the users of an open source project.

- They find gaps in the project documentation.
- They find bugs when deploying the software to their environment.
- They help with translating the user interface to their local language. End-user desktop applications are great examples of this (for example, Thunderbird, Firefox, and Inkscape).
- They are in a good position to help with user support in their native languages.
- They are good at identifying new features.

For the project, the goal at that stage should be to encourage interested users to contribute changes. In addition to making project communication transparent, there are several more options for lowering the bar for participation. For example, what is implicitly known to project members will be unknown to new contributors. It helps to make this implicit knowledge explicit.

- How exactly can the project be checked out of version control, modified, built, and tested?
- What are the project's preferred ways of communicating?
- How does one submit changes made to the project?
- What kind of turnaround time should contributors expect for changes to be reviewed, committed, and put into a new release?

It also helps to have really trivial "getting started" issues marked as such in the project issue tracker.[9] Several Apache projects explicitly label issues as suitable for new contributors.

It is also beneficial to actively engage with users, asking for bugs to be fixed or features to be added. Actively inviting them to become active and explaining that their contribution would be needed and appreciated helps with raising motivation to spend more time and effort beyond submitting an issue in the issue tracker.

SHARE CONTROL TO KEEP PEOPLE GOING

After getting a new contributor initially involved, the goal should be to bind them closer to the project. One way to do that long term is for contributors to turn into owners of the project. What are some options to do that?

A first step could be to invite new contributors to participate in activities that are typical for maintainers of an open source project, such as reviewing incoming changes and mentoring new people coming to the project. Another step to reward commitment to the project is to hand out commit access (that is, write access) to the project repository. There are also further steps that open source projects can take.

- Add new people to the group that decides who gets invited to the group next.
- Share ownership of project assets like trademarks, access to collaboration, and social media accounts.

> Making extensive documentation about the project transparently available also means that contributors can improve their own skills, thus directly feeding into mastery as an intrinsic motivator.

In a more informal way, it helps to explicitly mentor people and point them to growth areas. For example, if someone has been spending a lot of their time submitting bug reports related to the project documentation, inviting and helping them to directly improve that documentation can be a good next step. In that way, motivation grows as people notice that project participation helps them improve their own skills. When faced with growth challenges, Apache Beam established several best practices to level up people; one of those was to actively look for people who were likely good committers but needed some mentorship to actually make that transition. Their experiences were published in an informal way online.[10]

Another way of sharing control is detailed by Karl Fogel in his book, *Producing Open Source Software*, where he discusses delegating not only technical tasks but also management and coordination tasks.[11] One good example of that at the ASF is what has become a recommended practice for many projects—for coordination purposes, each project has one so-called project management committee (PMC) chair.[12] While, formally, this role comes with a lot of responsibilities, usually those are shared with the entire PMC group. In the past, though, it has happened that the PMC chair gained enough social credibility that, essentially, he/she was treated as the benevolent dictator of the project. To avoid this situation, a lot of Apache projects have adopted an informal rule to rotate the PMC chair role among group members (for example, on a yearly basis). That way, the likelihood of one individual acquiring too much influence becomes smaller.

PRAISE IN PUBLIC

Give people arguments to convince their employers. Much of the work on projects that are not being pushed forward by a well-funded entity is done by motivated individuals. Relying solely on altruistic motivation does bear the risk of participants burning out, in particular, if there is a imbalance between what participants are being paid for and what they are doing pro bono. This seems to be a major problem and one that we have yet to solve.

We still see examples at companies where reviews have, in fact, suffered negative consequences for employees making open source contributions because those were hours that (despite them being after-work hours) were not spent working on the intent of the business owner. So, from a project's perspective, it's a good idea to think of the motivation for employers to pay for open source contributions—particularly when the project is used in commercial contexts.

While there are many reasons to actively participate in projects, one of the more obvious reasons for companies to make time for employees to participate in projects is employer branding. (More often than not, large software shops use walled garden approaches to promote the appearance of this while not actually practicing it.) In turn, this provides an easy vector for projects. The more visible and public participation can be made, the more reasons there are for employees to spend time on these projects. Ways to publicly praise participants include mentions in commit messages and in release notes but also special mentions in issue trackers or additional information in press releases that go out for new software releases.

In addition, it helps to share information within the project about which setup and arguments for participation work well for decision makers. Two examples of such collections of arguments come from the financial sector,[13] as well as from members of the ASF itself.[14]

At the end of the day, though, a lot depends on the maturity of the employing company that is supposed to sponsor open source contributions. Often, contributing back comes at a later step than publishing an open source project. A similar observation was made by many individuals helping organizations tap into projects, one very prominent example being the European Union.[15]

Political-level institutions are starting to understand that using, developing, and supporting open source software in the public sector can have a direct positive impact on the local economy as more players are able to provide supporting services to make custom modifications. See also[16] for one campaign example that is fairly successful in the European Union as well as FOSDEM 2020[17] for specific examples of open source in the public sector in Paris and Baltimore.

In addition, the public sector has understood that the transparency that comes with open source software leads to more trust as well as a bigger chance for the interoperability of projects, even across country borders. Examples of this can be seen in advice from the European Union for the development of COVID-19 tracking apps[18] where, at least in Germany, the backing companies chose not only to release the code but conduct the development in the open as much as possible.[19] At the level of the United Nations, this can be seen in initiatives like the United Nations Technology Innovation Labs.[20]

Another way of teaching corporations the benefits of open governance models that are so common in open source projects is to let them adopt these practices to solve the in-house issues that they face (like siloed development). At InnerSource Commons (http://innersourcecommons.org/), corporate software developers can find a common language, set of patterns, learning path, and expert group to seek advice from when moving in-house software development toward a model that resembles open source software development. Often, after adopting this development model, the rate of upstream open source contributions increases as well.[21]

> Instead of being dependent on the original project authors when it comes to communication schedules, contributors can participate according to their own schedule.

WHEN COMMUNITIES FAIL

For projects that fail to survive for more than a couple of years, there's usually one of a few patterns that can be identified in retrospect.

Setting the bar too high

One reason for project failure may be setting the bar too high. Consider a wildly successful open source project that failed to pull in a diverse set of people. Instead of increasing the pool of people responsible for the project, the original maintainers set the bar for participation too high. As a result, the maintainers, overwhelmed by maintenance tasks, are more likely to burn out. Or, even worse, the project continues to see use long after the maintainers have stopped working on the code base, opening the door for security vulnerabilities and bugs to linger. Unfortunately, most open source projects don't notice these issues until it is too late.

Relying too much on one entity

Another way for projects to fail is to rely too much on one commercial player to provide financial backing for the majority of those active in the project. Several projects got into trouble after a single major sponsor pulled out. One lesson learned here is to diversify the group of contributors—and to actively seek support from multiple players, ideally those using the project in production.

Age

Lastly, the age of an open source project is a large factor in community involvement. Older projects naturally become more stable over time, thus requiring

fewer changes to fix found bugs and defects. Fewer changes, consequently, drive fewer contributors, despite the user base remaining quite large. Take the Apache HTTP Server Project, for example. It is ubiquitously used, yet it mainly has small, bug fix-style releases only. This lack of volume in changes can lead to fewer people being excited about starting contributions, despite the prestige of being included as a contributor.

While open source projects come in many different shapes, it is clear that building a community around any such projects needs to be a deliberate effort. It pays to have a good understanding of what motivates humans, how to moderate discussions, and how to facilitate conversations, even in tricky situations, when building those communities.

REFERENCES

1. "The Open Source Definition," Open Source Initiative. https://opensource.org/osd (accessed Mar. 10, 2021).
2. "Git rebase." Yarchive. https://yarchive.net/comp/linux/git_rebase.html (accessed Mar. 10, 2021).
3. *Dirk Hohndel in a Conversation With Linus Torvalds*. (Oct. 27, 2017). [Online Video]. Available: https://www.youtube.com/watch?v=NLQZzEvavGs&t=514s (accessed Mar. 10, 2021).
4. D. Riehle, "Single-vendor open source firms," *Computer*, vol. 53, pp. 68–72, Apr. 2020. doi: 10.1109/MC.2020.2969672.
5. "Open source archetypes: A framework for purposeful open source." May 2018. https://blog.mozilla.org/wp-content/uploads/2018/05/MZOTS_OS_Archetypes_report_ext_scr.pdf (accessed Mar. 10, 2021).
6. "Success at Apache: Remote collaboration in the time of coronavirus." The Apache Software Foundation, May 11, 2020. https://blogs.apache.org/foundation/entry/success-at-apache-asynchronous–decision (accessed Mar. 10, 2021).
7. "If it didn't happen on the mailing list, it didn't happen." The Apache Way. http://theapacheway.com/on-list/ (accessed Mar. 10, 2021).
8. I. Drost-Fromm, "Call to action—Mahout needs your help," https://lists.apache.org/thread.html/a6ac3b63ab6480a787150b01a26a1e15c6253e8356a77b05e844821e%401364119733%40%3Cdev.mahout.apache.org%3E (accessed Mar. 10, 2021).
9. "Export admin client metrics through stream threads," The Apache Software Foundation. https://issues.apache.org/jira/browse/KAFKA-6986?jql=labels%20in%20 (GoodForNew Contributors%2C%20 Newcomer) (accessed Mar. 10, 2021).
10. "An approach to community building from Apache Beam," The Apache Software Foundation, Feb. 22, 2019. https://blogs.apache.org/comdev/entry/ an-approach-to-community-building (accessed Mar. 10, 2021).
11. "Share management tasks as well as technical tasks," https://producingoss.com/en/share-management.html (accessed Mar. 11, 2020).
12. The Apache Software Foundation. [Online]. Available: https://www.apache.org/foundation/how-it-works.html#pmc for more background (accessed Mar. 11, 2020).
13. T. Langel, "Making the business case for *contributing* to open source." Opensource Strategy Forum 2018. https://www.slideshare.net/slideshow/embed_code/key/HyhXotpRbfE6ws (accessed Mar. 11, 2020).
14. B. Delacretaz. *How to Convince Your Left Brain (Or Manager) to Follow the Open Source Path Your Right Brain Desires*. (Mar. 6, 2020). [Online Video]. Available: https://www.youtube.com/watch?v=F0SmiQ3SF6Q (accessed Mar. 10, 2020).
15. The Apache Foundation. *#ACEU19: Thomas Gageik – Open Source Software at European Commission's Informatics Directorate*. (Oct. 23, 2019). [Online Video]. Available: https://youtu.be/2EvCF4XKLso) (accessed Mar. 10, 2021).
16. Public Money, Public Code. [Online]. Available: https://publiccode.eu (accessed Mar. 10, 2021).
17. "Organizing Open Source for cities: Adapting the Open Source Program Office," in *Proc. FOSDEM'20*, 2020. https://fosdem.org/2020/schedule/event/ospoforcities/ (accessed Mar. 10, 2020). Looks good to me; Date of access March 10, 2021
18. "Mobile applications to support contact tracing in the EU's fight against COVID-19: Common EU Toolbox for Member States, Version 1.0," eHealth Network, Brussels, Belgium, 2020. [Online]. Available: https://ec.europa.eu/health/sites/health/files/ehealth/docs/covid-19_apps_en.pdf (accessed Mar. 10, 2020). Looks good to me; visited March 10, 2021
19. "Corona-Warn-App," GitHub, San Francisco. [Online]. Available: https://github.com/corona-warn-app (accessed Mar. 10, 2020).
20. "Open source codes and the challenge of the SDGs: An UNTIL interview with Amanda Brock," United Nations Technology Innovation Labs. https://open-ae.eu/open-source-codes-and-the-challenge-of-the-sdgs-an-until-interview-with-amanda-brock/ (accessed Mar. 10, 2021).
21. GitHub, San Francisco. *Why Innersource is a Critical Component of FOSS Sustainability—GitHub Satellite 2020*. (May 7, 2020). [Online Video]. Available: https://www.youtube.com/ watch?v=Y_mjvrX7u-w (accessed Mar. 10, 2021).

ISABEL DROST-FROMM is a member of The Apache Software Foundation Wilmington, Delaware, 19801, USA. Contact her at isabel@apache.org.

ROB TOMPKINS is a member of The Apache Software Foundation, Wilmington, Delaware, 19801, USA. Contact him at chtompki@apache.org.