

Getting Started With Open Source Governance

Jeff McAffer, GitHub

Using and managing open source is essential in modern software development. Here we lay out a framework for thinking about open source engagement and highlight the key steps in getting started.

Open source continues to change the software landscape. More and more parts of our world are being driven (literally) by software, and more and more of that software is open source. This is fantastic in so many ways. In this article, I focus on getting started with governing open source use and contributions.

Governing your open source use is the right thing to do for your company and for the open source community. Your governance program will identify and mitigate security, legal, and community risks. It will also build up your team's confidence, enabling them to be more productive. At the same time, you will be more responsible about licensing

terms and more likely to participate in the project communities.

CREATING AN OPEN SOURCE PROGRAM

Adopting open source is a significant initiative and should be structured as such. You need a champion to drive the vision, an executive sponsor for air cover and resourcing,

and an explicit set of people and resources to deliver the effort. In short, you need an open source program.

The program's role is to make explicit the goals, policies, and mechanisms around open source. What are you trying to accomplish with open source? What are your biggest opportunities? Risks? What kind of talent do you need? Who makes the decisions? Setting up a program means answering those questions and more. Doing that in a coordinated way avoids duplication and ensures a more coherent and consistent approach across your organization. Such a program may be concentrated in one team, based in an open source programs office (OSPO), loosely structured as a virtual team, or, on a smaller scale, made up of passionate volunteers.

In many ways, open source adoption in an enterprise setting is more about changing culture than it is about



FROM THE EDITOR

Welcome back to the world of open source! Previous articles in this column reviewed open source licenses and obligations to observe when using open source in products. Realizing the complexity of such undertaking typically leads companies to the creation of an open source program office to govern the use of open source. In this article, Jeff McAffer, who started and ran the Microsoft Open Source Programs Office before recently moving on to work for GitHub, helps us understand what to do when setting up shop: that is, getting started with an open source program office. He provides both clear concepts and practical advice, and I expect (well, I know) that the next articles in this column will continue to help practitioners with putting open source to good use. Happy hacking! – *Dirk Riehle*

technology or licensing. Interacting with and depending on open source communities is very different than internal development and affects nearly every part of product development from planning to delivery. The open source program should identify where the company is now and where it wants to be, and it should plot a course to get there. Websites of the TODO Group (<https://todogroup.org>) and the Open Source Guides (<https://opensource.guide>) have many articles on how to think about and engage with open source from multiple points of view.

WHERE ARE YOU NOW?

To help structure your open source journey, it's useful to identify a few milestones along the way. There are a few different open source "engagement model" taxonomies out there. Allison Randal's blog "Capabilities for Open Source Innovation" (<https://allisonrandal.com/2017/11/25/capabilities-for-open-source-innovation-background>) is a good example. I put together an open source engagement model (<https://mcaffer.com/2019/02/Open-source-engagement>) based on the work I was involved in while transforming Microsoft to be an open source enterprise.

This model, summarized next, captures discrete states of open source

engagement. Most teams are simultaneously at multiple engagement stages. That's fine. There is also no "best state." Being proficient is more than adequate for many, while others seek to become masters. The point of the model is to help understand where you are and intentionally plot a course to where you want to be.

A great way to use the model is to test for each characteristic how the following phrases feel when applied in the context of your company:

- › "My team's approach to open source is _____"
- › "The value of open source to my team is _____"

Open Source Engagement Model

- › Denial
 - *Denial*: Somehow open source does not apply to your domain or is the wrong approach.
 - *Prevention*: Technical, legal/process, or regulatory barriers are put in place to block consideration of open source.
 - *Countering*: Open source is attacked with fear, uncertainty, and doubt.
- › Tolerant
 - *Limited*: Open source is used grudgingly and allowed only in pockets.

- *Experimental*: Open source is embraced by some early adopters deeply engaged in isolated areas; some releasing occurs.
- *Ad hoc*: Processes and policies are localized; the environment ranges from Wild West to locked down; outcomes are inconsistent.
- *Fearful*: Emphasis is placed on limiting risks; sequester teams are formed; engagements are tightly scoped.
- *Not rewarded*: No career incentives are offered for work related to open source; disincentives discourage "risky" behavior.
- › Hype
 - *Silver bullet*: Open source is going to transform the company!
 - *Marketing*: All the cool kids are doing it. We want to be cool, too.
 - *Recruit/retain*: Emphasis is placed on high-profile, high-volume "open source" hires.
 - *Incoherent*: Engagement is not coordinated or localized, or there are no policies/processes put in place.
- › Proficient
 - *Systematic*: Central policies and processes are developed around legal and security topics.
 - *Tooled*: Tooling is in place to track and guide open source engagement.
 - *Broad*: All teams are free to engage and understand the "rules of the road."
 - *Engaging*: Work with communities is encouraged; fixes/features are contributed.
 - *Efficient*: Open source is seen as a valuable tool to reduce time to market.
- › Fluent
 - *Value*: The business understands the value that use/

release does and does not bring.

- o *Fundamental*: The company bets on using or releasing open source to support core capabilities.
- o *Rationalized*: Policies and processes are continually reassessed and automated.
- o *Open*: Technical and process discussions default to open.
- o *Healthy*: Engagement health is integral to engineering/business reporting.
- o *Rewarded*: Open source engagement is explicitly recognized as a valued activity.

Make no mistake: changing the culture will be one of the biggest challenges.

› Mastery

- o *Integral*: Open source is integral to the business model from the beginning.
- o *Liberating*: The business understands the true value it adds and builds on that with open source.
- o *Disruption*: The business recognizes open source as a means of disrupting and quantum innovation.
- o *Shared control*: The company embraces the idea of open source in “coopetition” and for use in foundations, community initiatives, and joint projects.
- o *Proactive*: The company sees open source as a tool to engage broadly for improving quality and security.

WHERE DO YOU WANT TO BE?

Using this model, you can both understand where you are now in your open source engagement and where you want to be. What state matches your business goals? Without matching concrete, durable business goals, your open source program will waver and

evaporate. Open source is a long-term investment.

There isn’t a “best destination” here. It all depends on your goals, and your goals will change over time as will your target engagement state. The key is understanding the change and what it means. What concrete steps will you take to move from proficient to fluent or to make your engagement more rationalized? The model is here to help you choose your own adventure.

THE JOURNEY TO PROFICIENCY

Since this article is mostly about getting started, we’ll assume you are aspiring to

be proficient. Proficiency is all about setting up the virtuous pair of tasks: smoothing the path for open source engagement and starting to change the culture.

You can tool open source engagement all you want, but if your organization structure does not encourage or recognize it, your teams will not engage. Conversely, trying to get everyone to engage won’t work if the policies and mechanisms around engaging are cumbersome.

Culture

Driving the change to open source is like any other initiative: you need to tell your team what it is and why the change is needed, outline the opportunities and challenges that will likely come with the change, and talk about expectations. Investments here will pay huge dividends later both in satisfaction and in ease of adoption. Your culture initiatives should reflect where you are and where you want to be on the engagement spectrum.

Make no mistake: changing the culture will be one of the biggest challenges. Software is easy. People are hard. Overcommunicate. Engage all

stakeholders and constituents. Recognize that open source engagement goes far beyond libraries of code. For developers, it can strike at the core of their self-image as people who write the code. For managers, you are asking them to invite strangers into their systems and development processes. For executives, this change relinquishes some level of control over your business and forces (enables really) you to focus on your core business value.

Again, do not underestimate the challenges here. Champions and executive support are key.

Policy

Tooling open source governance starts with policy. Policy should be a reflection of the culture and operations you want to have. Policy codifies the opportunities you want to enjoy and risks you want to avoid. A good policy for open source use should be the following:

- › *Automatable*. Automatable policies focus on outcome or end state and leave out process and implementation details. A policy that requires “suitable confidence from legal” can be automated whereas “legal must sign off on uses” is not.
- › *Minimal*. Minimal policies avoid the many hypothetical risks and focus on what actually matters to your situation.
- › *Scenario driven*. Open source licensing terms vary by scenario as does business risk. Take this into account.
- › *Uniform*. Having many variations across the company confuses the teams, inhibits collaboration, and frustrates tooling efforts.
- › *Revised*. The industry changes; your business changes; your understanding changes. Iterate on your policies to remove friction, increase confidence, and maintain relevance.

Ensure the data your policy require are widely available and unambiguous

in both their definition and their use. Avoid duplicating data requirements and be sure to clearly identify plausible sources of the data needed. For example, don't ask developers to identify licenses. Those data are readily available, and developers generally don't understand licensing.

When putting together the policies, as with driving culture, include all of the stakeholders. Once you all understand you are on the same side, amazing things will happen. The key to that is getting on the same terms. By far, the most contentious and drawn-out discussions will be due to differing assumptions and terminology. After all, you have lawyers, developers, security folks, and businesspeople all trying to get on the same page. Take the time to nail down the nouns and verbs.

Inventory

Modern development involves using hundreds, if not thousands, of open source components. It's trivial to type "npm install" or "docker build" and get all manner of open source in your system, each with its own community, license(s), potential vulnerabilities, and so on. Detecting, tracking, and managing your inventory of open source are the core jobs of a governance system.

The automated detection of open source is essential. Even the most diligent development team will quickly tire of manually listing all of the open source they use. Keep in mind that it's not just your direct dependencies, it's all dependencies that you have to track and manage. A vulnerability or unfortunate license can appear anywhere in the dependency graph.

Anecdotally, it appears that manual reporting will track only 10–20% of components overall. Even if your team does manage to do the mind-numbing work of manually listing out each version of each component being used, as soon as they finish, and with zero changes on their side, a new "install/restore/update" could change the graph significantly.

The mechanisms for detecting the open source you use vary by ecosystem. NPM

users may be able to simply observe their package-lock.json file. Maven, NuGet, and Go users have a bit more work to do. There are a number of open source and commercial solutions for this. If you are on GitHub, various types of dependencies found in your repositories are detected and surfaced for you. Either way, get a tool/system and use it. Your developers will love you for it. Open source governance tooling will be covered in a future column.

Data and Insights

Knowing what open source you are using is half the battle. The other half is

to get compliance and project information. GitHub recently announced support for maintainer security advisories (<https://help.github.com/en/articles/about-maintainer-security-advisories>) in a bid to increase reporting on vulnerabilities.

Community health is a nascent field with ongoing work best typified by the CHAOSS project (<https://chaoss.community/>). That project has identified metrics you can use to assess the diversity, inclusivity, evolution, risk, and value of open source projects. One metric, for example, is the bus factor, "the minimum number of team members

Modern development involves using hundreds, if not thousands, of open source components.

understanding the nature of the components. What licenses are involved? Do they have security vulnerabilities? Who's behind the projects? Are the communities healthy? These data enable you to make decisions about the open source you use. Having quality, machine-readable data enables automated policy evaluation.

Unfortunately, there is very little consistency to the availability, accuracy, or form of the data today. Only an estimated 60% of vulnerabilities are noted in the common public vulnerability databases (<https://res.cloudinary.com/snyk/image/upload/v1551172581/The-State-Of-Open-Source-Security-Report-2019-Snyk.pdf>). About 35% of Maven packages have missing or ambiguous licensing (<https://clearlydefined.io/stats>). Those are pretty big gaps.

Often the toolset you use to get inventory will also have licensing and vulnerability data that will, at least partially, fill these gaps. That's great. There are other freely available sources of data. For example, ClearlyDefined (<https://clearlydefined.io>), a crowd-sourced effort to clarify open source project data, is a good place

that have to suddenly disappear from a project before the project stalls due to lack of knowledgeable or competent personnel" (https://en.wikipedia.org/wiki/Bus_factor).

These data and insights help you make smart choices and manage ongoing engagement. They answer questions about policies and help identify areas where your program is at risk and where you can best contribute to the ecosystem on which you depend.

Automation

The policy, inventory, and data discussed come together as the basis for automation. Automation is the key to smoothing the process and generating the confidence you need to know you are "doing it right" for the situation at hand. That last part is important: the situation at hand.

Many policy choices are highly dependent on the scenario. For example, different considerations are required depending on whether you are contributing to a project with a contributor license agreement or a developer certificate of origin, using certain licenses in distributed versus cloud software,

or using vulnerable components internally rather than in a client application. Your automation should apply your policy to the inventory using the data and scenario to determine the outcome.

Done right, governance automation should be entirely invisible to your teams until something unusual happens. For example, with a powerful policy, accurate inventory, and high-quality data, the Microsoft OSPO team manages millions of open source uses with less than 1% requiring human interaction.

Many open source and commercial tool sets are available to help you drive this type of automation and become proficient. As with the inventory automation, get one and use it. You'll be thankful.


BEYOND PROFICIENT

This discussion has largely focused on open source use. With use naturally

comes the contribution of bug and documentation fixes and new features or design ideas. Treating the open source you use as though it were your own code is a sure sign of proficient engagement.

Another way to start using open source is through corporate acquisitions. Just as your products are increasingly based on open source, so too are those of would-be acquisitions. When you buy a company, you are buying its governance. Use the model and concepts here to compare the target's governance to yours and look at how its policies were implemented. That will tell you a lot about how well the company's software fits into your world. For potential targets, bear in mind that acquiring companies will want to know all this information and that retroactive discovery is way more expensive. Not only is proactive governance

the right thing to do, it will make you a more attractive target.

Releasing your own code as open source is also a natural progression. The concerns here are somewhat different, but releasing too requires policies, tooling, and data. For example, your policy should address processes for approving the release of intellectual property, managing patents, licensing, handling community development and governance, reviewing business goals, and more. You should have tools that manage the creation and structure of public repositories, and insights that measure community reach and engagement. But these are topics for another article. 

JEFF MCAFFER is senior director of product at GitHub. Contact him at jeffmcaffer@github.com.

**SUBMIT
TODAY**

IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING

► SUBSCRIBE AND SUBMIT

For more information on paper submission, featured articles, calls for papers, and subscription links visit: www.computer.org/tsusc



Digital Object Identifier 10.1109/MC.2019.2937706