

The Future of the Open Source Definition

Dirk Riehle , Friedrich-Alexander-Universität Erlangen-Nürnberg

Many forces pull to change the definitions of what free and open source software are. This article looks at these forces and speculates what the future will hold in store for the definition of open source software.

The definitions of what free software and open source software are use different words but are essentially the same. Both free and open source software can summarily be defined as follows:

“Free and open source software is software that is available under a license that grants everyone the right to use the software, to modify the software, and to pass on the software to third parties, modified or not, all for free.”

The availability of source code is implied by the ability to modify the software to one’s liking.

Digital Object Identifier 10.1109/MC.2023.3311648
Date of current version: 13 November 2023

BASE DIMENSIONS

The free software definition was first written in 1986 by Richard Stallman for the Free Software Foundation (<https://www.gnu.org/philosophy/free-sw.en.html#fs-definition>). It defines the four “essential” freedoms of software. In 1998, the newly

founded Open Source Initiative (OSI) defined open source software using a ten-item bullet list of criteria that a license must fulfill to be considered an open source license (<https://opensource.org/osd/>).

In an earlier instance of this column, Jesus M. Gonzalez-Barahona provides an excellent overview of the history of free/libre and open source software.¹

For this article, two aspects are notable about the open source definition:

1. It does not say anything about the development process.
2. It does not restrict the use of the software in any way.

In the remainder of this article, I will use the term open source software to include all variations that the various

FROM THE EDITOR

There is turmoil in open source land. An increasing number of software companies that provided some or all of their products as open source software have stopped doing so. They have switched away from open source to alternative licenses. In this column, I take a look at why this is so and how these and other events have led us to review the very fundament on which open source software rests, the open source definition. Happy reading everyone, and keep on hacking!—Dirk Riehle

communities use: free software, free/libre software, free/libre and open source software, etc.

USE OF THE DEFINITION

The website of the OSI provides us with the definition of open source software in a formal and structured way, as a list of ten bullet items. The OSI also operates the license-review and license-discuss mailing lists, which serve as the arbiter and decider of whether a particular software license is an open source license or not. Any license that passes the review will be added to the public list of open source licenses on the OSI's website.

When reviewing a software license for inclusion in the approved open source license list, those who argue regularly go back to the open source definition and compare and evaluate the proposed license against its ten criteria. The open source definition this way serves as a specification. It has proved its value time and again as a practical and sharp tool for making decisions. I view the definition as a significant cultural achievement.

Showing up as an open source license on the OSI's website constitutes a stamp of approval of the license, bestowing the goodwill that comes with the term "open source" on any software provided under this license. Many organizations have tried to get licenses they created (for their own purposes) approved as open source licenses and failed.

OPEN PROJECT GOVERNANCE

One important aspect that the open source definition does not address is how open source projects are governed. Governance consists of the practices and processes of how the project operates, how decisions are made, who can contribute, etc. The open source definition is solely about software, the artifact. It does not mention how the project conducts its business, that is, how the software is being developed.

Still, the open source development processes were on the minds of the founders of the OSI. For most of its lifetime, and still today, another section on the OSI's website has this to say about open source software (<https://opensource.org/about/>):

"Open source enables a development method for software that harnesses the power of distributed peer review and transparency of process. The promise of open source is higher quality, better reliability, greater flexibility, lower cost, and an end to predatory vendor lock-in."

This paragraph clearly states that in the authors' minds, open source software is developed in a communal way. For most of open source software's early life, this was the case. Most important open source software was and is being developed in an open way, following the principles of open collaboration: everyone can participate, decisions are made based on the

merits of arguments, and participants decide about their own processes (<https://dirkriehle.com/ocd/>).

Every project is different, but there are clearly established patterns as well, be it the peer group model explored by the original Apache web server team or the benevolent dictator for life model explored by Linus Torvalds.

Open source, the artifact, and open source, the governance process, create the two dimensions by which we can classify software development projects. Figure 1 displays this 2 × 2 matrix.

The first two types of software development projects are as follows:

1. Community open source is open source software (by license) that is being developed in an open and transparent way following the principles of open collaboration.
2. Tightly controlled open source, here reduced to vendor-owned open source, is open source by license developed in an intransparent process, for example, behind the closed doors of a vendor.

The two remaining types of software development projects are proprietary, by license, and owned by the development organization.

3. Inner source is proprietary software developed using the principles of open collaboration within an organization.
4. Closed source software is the traditional proprietary software developed using established engineering processes like Waterfall or Scrum.

VENDOR-OWNED OPEN SOURCE

In the early noughties, software vendors discovered that open source software is a great method for getting a

foot in the customer's door. Making their product available as open source software affords them frictionless distribution. In a past instance of this column, I explained how vendors turn "free-loading users" into "paying customers."²

The vendors of vendor-owned open source are traditional software vendors like Elastic, MongoDB, or HashiCorp and not open source distributors like Suse, Red Hat, or Univention, which typically do not own the open source code they are building their distributions from.

The owner (copyright holder) of some software can license out the software under one or more licenses. A vendor can make the same software available under an open source license and under a commercial license. To be able to sell a commercial license to their software, a vendor cannot allow that their rights to the software get diluted.

For this reason, most vendors require that any potential contributor to the open source software sign over the rights to their contribution to the vendor. The required type of contract is called a contributor license agreement (CLA). CLAs are used by an organization to centralize all needed rights to the software, for example, to represent it in court. They were originally invented by the open source foundations but like many legal tools are now used by vendors as well.

Foundations and vendors use CLAs differently, though. While foundations do not take private contributions, vendors can (and do).

Not surprisingly, the practice of acquiring copyright to tightly control it is disenchanting to developers. Therefore, vendors typically do not receive many contributions and therefore develop most if not all of their code themselves. They do not do so in the open but keep their road maps secret to hinder competition. In contrast to open source projects run under an open source foundation, the governance of projects run by a vendor is typically closed, not open.

STRENGTHENING THE DEFINITION

Both the closed governance and the copyright assignment are bothersome, even infuriating to true open source enthusiasts. Vendor-owned open source has been called "faux-pen source software" (fake open source software) or simply just a fraud (<https://meshedinsights.com/2021/02/02/rights-ratchet/>).

As a consequence, many such enthusiasts have repeatedly asked that open source governance be added to the open source definition. Not just the artifact but also the development process should be open, before some software should be called open source software, and nobody should centralize the rights to the software in one controlled place.

These calls for extending the definition have led nowhere. Obviously, the vendors opposed it, and they are clearly members of the community. Open source software development today is mostly paid for by companies, so it has become vendor friendly, and attacking business strategies through a changed definition of what constitutes open source has received little support.

I think that it would be quite hard to come up with actionable definitions of what makes a governance process open or not. It may not be impossible, though: The Apache Way is a codification of some process practices that could lead the way to an open source definition that includes open

governance as a key aspect (<https://www.apache.org/theapacheway/>).

I do not hold my breath, though, for an extension of the definition to include open governance. Things are too established with too many invested forces for anything to change quickly.

DISCRIMINATORY LICENSES

A key aspect of the open source definition is that it does not discriminate against specific uses or parties.

There is plenty of people-killing machinery like rockets and drones that run Linux. This is fine by the open source definition! It is not fine by some software developers. For this reason, Coraline Ada Ehmke founded the Organization for Ethical Source, which tries to mirror the OSI in defining and approving so-called ethical licenses (<https://ethicalsource.dev/>). Ethical licenses encode the authors' value system most notably by disallowing specific uses. This then discriminates against these uses and by definition makes ethical licenses not open source licenses.

Unrelated, but in a similar way, vendors with a vendor-owned open source strategy always want to prevent anyone else from competing with them using their own software. For most of the early life of a product, open sourcing using an aggressive copyleft license and providing a separate commercial license did the trick and kept the competition away.

This changed, however, when vendor-owned open source became popular and the large cloud service providers

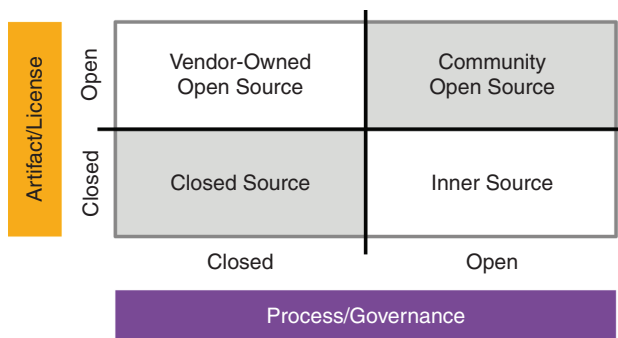


FIGURE 1. The classification of software development projects.

(Amazon Web Services, Microsoft Azure, and Google Cloud) started providing the vendor's open source software as a cloud service. The existing open source licensing strategies were not enough to keep the hyperscalers away. According to the vendors, competition by the hyperscalers is unfair and needs to be prevented.

Examples of vendors that relicensed from open source to source available are MariaDB, MongoDB, and Elastic and, more recently, Akka, HashiCorp, and Cockroach Labs.

These source available licenses sometimes put in an effort to make the license more palatable. For example, some source available licenses revert

As the saying goes, vendors will have to pry the open source definition from the OSI's cold dead hands, and that would be such a pyrrhic victory that I do not expect it to happen.

To stop the hyperscalers (and anyone else) from competing with them, these vendors invented so-called source available licenses, also known as non-compete licenses. These licenses basically say that the software is like open source, unless you want to compete with the vendor, in which case you are not allowed to use the software (hence non-compete). Obviously, by the open source definition, source available licenses are not open source licenses.

Vendors license away from open source to source available typically only if they feel they need the goodwill of open source less urgently than before. As a business, these vendors' products probably matured and already reached the channel-product fit.

to the venerable Apache license for code that is older than two years. Still, its discriminatory nature remains.

I consider the relicensing of a product that is available as open source software by its vendor to a source available non-compete license a foregone conclusion. Once a product matures, for example, by reaching the channel-product fit, the open source strategy will lose some significance, and the need to increase profitability and return on investment for the venture capitalists behind the vendors will take over.

As Figure 2 shows, only community open source, if run well, will stay open and transparent. Vendor-owned open source will get less and less open over time.

WEAKENING THE DEFINITION

Open source has matured; it is now a household name that carries a good, even a sterling reputation. Open source is good! Companies and private users alike love using high-quality open source software! The governments of the world increasingly are pushing for open source in public tenders!

Not surprisingly, vendors like the goodwill that open source bestows on them, even if some in the open source community consider their work fake open source.

As a consequence, the drumbeat of public articles, vendor blog posts, and conference presentations has been increasing to weaken the definition of what constitutes open source software. Proponents are asking to revise the definition to include, most notably, source available licenses.

While the pressure is mounting, I see no wavering in the OSI's stance to not accept any discriminatory language in the open source definition, and I am very happy about this. As the saying goes, vendors will have to pry the open source definition from the OSI's cold dead hands, and that would be such a pyrrhic victory that I do not expect it to happen.

There are enough other vendors who benefit from community open source to oppose those vendor-owned open source firms that would temporarily benefit from weakening the open source definition.

STEMMING THE TRUST EROSION

Some argue that vendors licensing away from an open source to a discriminatory license have been eroding the trust in open source. I beg to differ. As explained, open governance and community participation were never part of the original open source definition, and this has been obvious in these vendors' behavior.

It is a vendor's prerogative to define and execute their business strategy. Nobody can tell them how to go about it. CLAs and lack of community

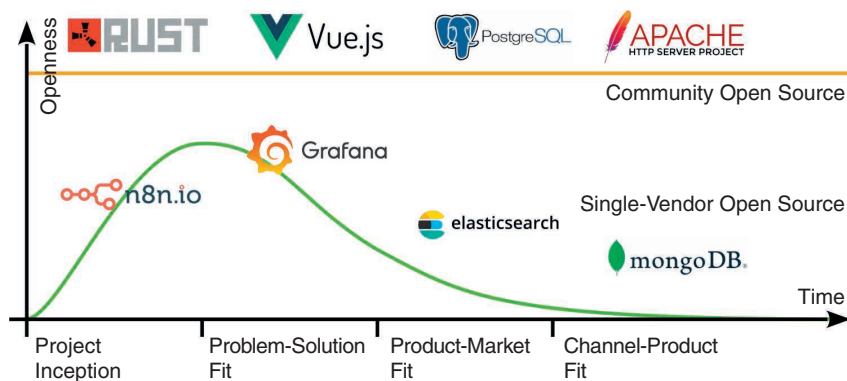


FIGURE 2. The openness over time of community versus vendor-owned open source.

participation are clear signs of commercial intentions to acquire a superior return on investment like traditional closed source vendors could have.


Anyone who uses open source, whether for in-house use only or as a component in products, creates a dependency on this open source software. Such dependencies need to be thought through. Thus, in my book, it was always clear that vendors would eventually try to tighten the screws. Anyone who uses vendor-owned open source needs to recognize that payday will come, sooner or later.

Thus, users need to think through their use of open source software. What is the impact of adding a specific dependency? If the conclusion is to use vendor-owned open source, fine! If not, fine as well!

OPEN SOURCE, WHAT NEXT?

The vendor-owned open source firms will not succeed in weakening the open source definition. I expect them to move on and utilize secondary devices to bestow goodwill onto their products. An obvious choice for a secondary device is open source foundations. I expect real or fake foundations with the goal of channeling goodwill and marketing attention to the products of the commercial firms backing them.

The open source world at large would be well advised to come up with their own commercial open source foundation. The essence of the foundation would be to provide independent specifications and certifications of good behavior. Certification marks could be acquired (and lost) by vendors who seek such

certifications to win user trust. This way, open source would keep its sterling reputation. 

REFERENCES

1. J. M. Gonzalez-Barahona, "A brief history of free, open source software and its communities," *Computer*, vol. 54, no. 2, pp. 75–79, Feb. 2021, doi: 10.1109/MC.2020.3041887.
2. D. Riehle, "Single-vendor open source firms," *Computer*, vol. 53, no. 4, pp. 68–72, Apr. 2020, doi: 10.1109/MC.2020.2969672.

DIRK RIEHLE is a professor of open source software at Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany. Contact him at dirk@riehle.org.

Over the Rainbow: 21st Century Security & Privacy Podcast

Tune in with security leaders of academia, industry, and government.



Bob Blakley



Lorrie Cranor

Subscribe Today

www.computer.org/over-the-rainbow-podcast