DETERMINING THE TRUE OPENNESS OF OPEN SOURCE PROJECTS

IBRAHIM HADDAD, PHD

THE LINUX FOUNDATION

Ibrahim Haddad, PhD

Determining the True Openness of Open Source Projects

The motivation for writing this paper originated from various discussions evolving around what makes a project a true open source project beyond just the choice of license. People have different opinions and thoughts about the various indicators of a project's openness. In this paper, we explore such indicators that together can help define the true openness of a given project and conclude with some recommended practices and other practices to avoid in an open source project, touching on a dozen different areas. We hope this paper becomes a trigger for new conversations in open source projects on how to be more open, transparent, and inclusive.

Printed in the United States of America

First Edition, 2019

1 Letterman Drive
Building D
Suite D4700
San Francisco CA 94129

Contents

Introduction

The success of an open source project depends on many factors, where openness is one of the essential ones. The primary premise of this paper is to explore and identify the various indicators that can provide insights about the openness of a given project. The paper is organized in four sections:

- **Openness indicators:** This section examines such indicators and discusses how they contribute to the openness of the overall project.
- **Best and worst practices:** This section provides recommendations for practices that can enable and foster an open environment that will help open source projects grow and prosper. The section also covers some of the worst practices that you want to absolutely avoid.
- **Characteristics of a great open source community:** This section offers thoughts on the common characteristics of successful and thriving open source project communities.
- **Call to action:** This section focuses on how participants in open source projects can do a better job with respect to openness.

Chapter 1

OPENNESS INDICATORS

GOVERNANCE

Governance determines who has influence and control over the project beyond what is legally required in the open source license. A project's governance model establishes a collaboration framework that addresses difficult questions such as:

Contributions

- Who makes decisions for code inclusion and releases, and how?
- Who can be the lead maintainer or architect for the project (larger projects have more than one)?
- How can the project contributors become maintainers or committers?

Direction and Finance

- How can the project raise money and who decides how this money is spent?
- Should the project have a Technical Steering Committee or a Conformance and Certification Committee? Who can be on them?
- Who decides the project's direction and roadmap?

Transparency

- Who can participate in the discussions and decide on critical matters?
- How transparent are the decision-making processes?
- Can anyone follow the discussions and meetings that take place in the project?

Re-use

- What compliance requirements are there for redistributing, modifying or using the software?
- How can the project enable contributors and downstream re-distributors to comply with these requirements?

Copyright and trademark

- Who owns the copyright on contributed code?
- How can users license the project's branding?

ACCESS

A key indication of project openness is how publicly accessible the project's resources, communications (mailing lists, IRC, Slack, etc.) and history are, beyond the current active participants. For starters, an open project will provide the same level of source code availability to all developers, meaning there is no favoritism to developers via priority access.

Collaboration in most open source communities is centered on a relatively standardized set of tools, such as wikis, IRC, and mailing lists, which allow members of the community to communicate with each other. It is worth noting though that there may be circumstances where mailing lists with limited distribution are appropriate, e.g. for handling pre-disclosure security vulnerability reports, however, these are rare and special cases. Open source communities often rely on tools such as GitHub, git, Bugzilla, JIRA, and file servers to collaborate on code development; wikis and blogs are often used to inform about the community efforts. Project policies and infrastructure must be in place to ensure developers can adequately interact with each other using these tools.

Additionally, open projects provide access to developer tools such as mailing lists, forums, bug-tracking systems, source code repositories, and documentation. Participants are able to join discussion platforms, decision-making mechanisms, and project roadmaps so it is possible to understand why and how the project makes decisions.

PROCESSES

A project with a high degree of openness will have clearly defined processes for how things work in the community and how to contribute to the project. For starters, a clear development process should outline how to incorporate code into the project, the release process and schedule of the project, and any requirements developers need to meet to get their code accepted. This should also include guidelines for participation that demonstrate community best practices for things like patch submissions, feature requests, bug reports, and signing-off on code contributions.

DEVELOPMENT

An open development process enables developers to influence the direction of the project via contributions. It encourages contributions through the visible recognition of the developers and the provision of a transparent contribution and acceptance process that provides clear feedback on updates to contributions as they are incorporated into the project. This transparency should also allow external participants to identify the source from which code contributions originated.

Release early and release often is a practice that has been integral to open source software for most of its history. This practice allows open source communities to innovate at a rapid pace with a high quality of code because it creates a much faster feedback loop between developers, testers, and users. Releasing early allows feedback at an earlier stage of development so new ideas can be incorporated while the code is still flexible; it also allows any potential issues to be flagged more quickly. Releasing often results in smaller changes that are easier to understand, debug, and improve which makes it much easier to maintain a rapid development pace. This practice also aligns well with the progressive movement of many industry projects towards agile development and continuous integration / continuous delivery (CI/CD) methodologies.

COMMUNITY STRUCTURE

Open source project communities usually start with a flat structure and transition to a hierarchical structure as they grow in terms of contributors, and as the body of code becomes more complex, requiring additional maintainers. From that perspective, the code leadership evolves around committers, maintainers, and reviewers (please note that not all projects support these levels of contributors).

Two key factors that indicate the openness of a project's community structure are:

1. The commitment that individuals responsible for the project leadership get their roles based on talent, effort, and achievements in the project.

2. A key component of community openness is the accessibility to become a committer, reviewer, or maintainer. This process should be clearly documented and equitable so that any contributors to projects have the potential to be promoted to one of these roles.

RELEASE NOTES

In open source projects, with hundreds and possibly thousands of developers, documenting releases is a fundamental requirement. There are many advantages that result from providing detailed release notes, such as providing visibility into the project's progress, documenting continuous improvements to the project with every release, providing a great reference for new users of developers joining the project, and in general using it as a communication tool.

Another possible related openness indicator is how the project offers credit to all contributors via the release notes or a specific file that lists all contributors.

ROADMAP

At a high level, roadmaps provide high-level overviews of the project's goals and deliverables for that release. Open source projects that maintain an open roadmap achieve several advantages and are able to:

- Communicate the plans and goals for each release (minor, major, etc.),
- Manage the expectations of its users and developers by generating a shared understanding across everyone involved in the project, and
- Expose the project's plans to other open source projects that possibly rely upon or use them as a dependency.

LICENSE AND INTELLECTUAL PROPERTY CONSIDERATIONS

License

The license of an open source project determines the rights to use, copy, modify, and distribute the code. The choice of license for an open source project is an essential factor in determining the openness of the project. Open source projects should only use licenses that are approved by the **Open Source Initiative** and/or recognized as "free / libre" by the Free Software Foundation. Such licenses allow software to be freely used, modified, and shared. To be approved by the Open Source Initiative, a license must go through their l**icense review process** to confirm that the license satisfies their **Open Source Definition** ("OSD"). You may come across many other licenses that are incompatible with the OSD. Most of these licenses are considered "Source Available" licenses that commonly include restrictions or limitations on the use and/or distribution of the software. These restrictions often render the licenses as incompatible with the OSD.

Derivatives

Developers should be able to create and distribute derivatives of the source code for their own projects or reuse the code in other projects. To allow this, the project needs to be available under an appropriate license that provides these freedoms.

Contribution mechanisms

A key consideration for any project is the mechanism by which they manage the provenance of incoming code contributions. Open source projects deal with these concerns differently. Some projects adopted a developer certificate of origin, others require a contributor license agreement, while many projects (particularly smaller ones) do not use formal contribution provenance mechanisms.

DCO sign-off process

The Developer Certificate of Origin (DCO) sign-off process ensures that every single line of code accepted into a project has a clear audit trail. It is a developer's certification that they have the right to submit code for inclusion into the project. The Linux kernel process for instance requires all contributors to sign-off their code, which indicates the contributor certifies the code as outlined in the **Developer Certificate of Origin**. The signature communicates that the contributor has created or received the contribution under an appropriate open source license that allows it to be incorporated into the project's code base under the license indicated in the file. The DCO establishes a chain of people who take responsibility for the licensing and provenance of contributions to the project.

Contributor license agreement (CLA)

Some projects require either developers or their employers signing a CLA. Unlike the DCO, the text of CLAs can vary significantly from project to project, so the terms of any given CLA may have different effects. The purpose of a CLA is to ensure that the guardian of a project's outputs has the necessary ownership or grants of rights over all contributions to allow them to distribute under the chosen license. In some cases, this even means that the contributor will grant an irrevocable license, which allows the project to distribute the contribution as part of the project.

Software Package Data Exchange license format

Many of these open source projects have code licensed under different

licenses. Some projects are already adopting the **SPDX** format as a method to communicate the license information. One openness indicator could be how well a project makes explicit the various licenses for its different pieces of code via the standardized SPDX short-form license identifiers in every file. Additionally, a project can provide detailed license, copyright and other related information in a standardized, open, human-readable and machine-readable format by providing a bill of materials as an SPDX document.

DOCUMENTATION

An open source project can provide different types of documentation to help both users and developers of its community. Historically, documentation has been an area that is lacking and requires improvements. However, this is changing and many of the projects, especially those hosted within an open source foundation, have great documentation that cover all areas of the projects. In the following subsections, we examine three core areas where documentation is essential.

- Project
    - Mission
    - Governance
    - Community structure
    - Release cadence
    - Roadmap and priorities
    - Use cases
    - FAQs

- Documentation targeted for users:
    - User guide and tutorials
    - API guide
    - Architecture overview
    - Installation guide
    - Feature request process
    - Experience sharing section

- Documentation targeted for developers:
    - Detailed architecture and mapping to code sub-systems/ services when applicable

- Development process
- How to get involved
- Guidelines for participation
- Feature request process
- Patch submission process
- Signed-off-by process, when applicable
- Developer guides and tutorials
- API guide

Chapter 2

RECOMMENDED BEST PRACTICES

In this chapter, we highlight some of the recommended practices that support and enable open source projects, and also provide some practices to avoid.

	Recommended Practices	Practices to Avoid
License	OSI-approved open source license or FSF free/libre license.	• No license. • Unclear or conflicting licensing terms. • Vanity license. • Create a new license.
Governance	A governance model that gives equal footing to all current and future contributors to the project. Open source projects with an open and transparent governance model have better chances to grow, have a healthy environment, and attract developers and adoptees.	• No governance. • Biased governance that is dominated by a given party, usually the founder of the project.
Access	• Project resources are accessible to any users or developers interested in the project. • Anyone can participate in the project. • Any participant can earn committer rights by way of contribution and build trust with the project's community.	• Limited access based on sponsorship level or other factors.
Processes	• Documented processes for requesting a feature, reporting bugs, submitting code, etc. • Code is only committed through the project's defined process for incoming contributions. • All code goes through a peer review process.	• Ad-hoc or poorly designed processes. • Processes that keep changing or are stale and need improvements in order to scale and accommodate the development status of the project. • Processes that are not followed or respected.

	Recommended Practices	Practices to Avoid
Processes (Continued)	• The process to become a committer / maintainer / reviewer is enforced by the project for consistency. • The project's community revises its processes based on incoming feedback to ensure they continue to meet the project's needs as it grows and scales.	
Development	• Responsibility for development allocated to the individuals with the best capacity to deliver. • The project enforces quality standards when merging code. • The project implements multiple levels of review before entering final release. • Peer review is mandatory and public.	• Peer review is not enforced. • Pedigree of incoming code is not verified. • Project does not have a sign-off process or equivalent. • Contributors do not follow sign-off process while the code is still merged.
Community	• Accessible to newcomers - open development generally strives for inclusiveness. • Focused on visibility with emphasis on open decision-making processes and communication. • Self-organizing where individuals contribute in their areas of interest, or those of their employers. • Resilient to organizational change given that leadership is earned with experience. If individuals cease to participate, there are others to take their place.	• Little or no help or support available to new developers entering the project in terms of guidance, documentation, and mentorship. • Obscure decision-making process.
Community Structure	• Meritocracy drives advancement and acceptance. Contributors who provide the most value to the community are granted project leadership roles. • The project welcomes newcomers who have freedom and access to participate in public discussions, development, and testing.	• Structure biased towards a certain company, coalition, or commercial interests. • No clear path for developers on how to be promoted to a committer, reviewer, or maintainer.

	Recommended Practices	Practices to Avoid
Community Structure (Continued)	• The project's hierarchy is scalable because it consists of maintainers who oversee specific bodies of code in levels that can be added or removed as needed based on the size of the community. • Anyone can submit patches, and both developers and users are involved in the testing process. The roles of developer and user are closely integrated in open source development, allowing users to have a more direct path to influencing the project.	
Releases	• To protect certain users from the instability of rapidly developing software, projects provide stable releases that restrict the addition of experimental features to provide a reliable version that better supports use cases that rely on stability. • Weekly or monthly stable releases provide users and developers with the newest functionality after it has been tested • Long-term stable versions extend to longer periods and often only include security patches and bug fixes.	• Unclear structure of releases and branches. • Undocumented release processes. • Documented processes but uncommunicated and/or hard to locate on the wiki or the web site of the project.
Release Cadence	• The project has a defined cadence for its releases with set goals per release. • The release cadence and the goals to be met by each release are known to all projects stakeholders.	• No release cadence • Cadence is not suitable or does not meet the needs of the end users.
Derivatives	Open source license provides the freedom to create and distribute derivatives.	Non-OSI approved license or non FSF free/libre license that limits these freedoms.
Communication tools	Such tools include mailing lists and IRC, among others, and are available and open to anyone wishing to participate in the project.	• Restricted access to some of the communication tools. • Discussions happening in private chat rooms or private mailing lists.

	Recommended Practices	Practices to Avoid
Transparency	Open source communities must be as transparent as possible to attract new participation. • Contribution transparency. • Peer review transparency. • Transparency of discussions. • Transparency of promotion to committer or maintainer.	• Ambiguous decision-making process. • Favoritism in code acceptance based on origin and not quality of code and result of peer review. • Discussions with direct impact on project (architecture, development) happen in private with some rare exceptions of communication that for instance relate to the distribution of pre-disclosure security vulnerabilities.
Development tools	Available and open to all.	• Limited access. • Dependencies on proprietary tools prohibiting non-corporate contributors from participating in the development efforts.
Documentation	Availability of documentation covering architecture, APIs, installation guides, developer guides, development processes, participation guides, tutorials, etc.	• No documentation (source code is documentation) • Poor documentation. • Unmaintained documentation.

Chapter 3

CHARACTERISTICS OF A GREAT OPEN SOURCE COMMUNITY

Great open source communities may differ in what they work on and how they implement the structure and processes of their projects, but they share several characteristics:

- Community members work together for a common goal with a high sense of cooperation.
- Project participants feel free to express their opinions, share their ideas, and engage with other project members.
- Community members chose their maintainers and committers based on their expertise, level of contributions, and thought leadership. The community maintains a clear process for the selection criteria.
- The project's community is accessible to newcomers as users of the project or developers who wish to participate and contribute. Open development strives for inclusiveness.
- Great open source communities are very transparent with a strong emphasis on open decision-making processes and communication.
- Great open source communities are resilient to organizational change. Leadership is earned with experience and with the approval and consensus from community members. If individuals cease to participate in the project, there are others to take their role with minimal disruptions to the project and a clear process to guide the selection of the new leaders (maintainers).
- Great open source communities work to ensure that those who fall in minority populations are not treated differently. These communities give a voice to minority populations through frequent

consultation with members of those societies about how the community can improve to meet their needs better.

- Great communities do not limit contributions to just code and offer a wide range of contribution opportunities for non-coders in areas such as testing, documentation, communication, marketing efforts, and many more.
- Great open source communities foster a feeling of connection and collaboration among its members by providing plenty of opportunities for interaction. They create a feeling of connection that makes members more motivated to work towards the projects' goals.

A healthy and strong open source community is inclusive and diverse. Many open source projects are working to increase their inclusiveness, the diversity of their contributors, and to encourage new participation.

Chapter 4

CALL TO ACTION

This chapter focuses on the question of how we can do a better job with respect to openness. Three primary players come to mind:

- Open source developers – create new open source projects, contribute to existing projects.
- Open source leadership – on behalf of their company, they encourage and support the participation of internal engineers to open source collaborative projects; they support stakeholders and compliance teams in decisions to open source internal code; they foster discussion with their peers at other companies; they investigate opportunities to create new open source projects and collaborations.
- Open source foundations – such foundations host open source projects within a neutral forum, create new open source projects in support of their members, mentor developers, advise projects on policy issues, etc.

We believe these three key roles are instrumental in shaping the openness on any open source project. In the following table, we identify some of the actions these players can exert in the various areas that would help an open source project get to a higher level of openness.

	Open Source Developer	Open source Leadership	Open Source Foundation
License	• Avoid projects with vanity or unclear licenses. • Choose an OSI-approved license for their own project(s). • Communicate the benefits of using an OSI-approved open source license to colleagues. • Understand the license choice of your project or the license of the project(s) you want to participate in.	• Open source code using OSI-approved licenses only. • Mentor company executive on the adoption hurdles a vanity license poses.	• Educate hosted projects on the right choice of license for their projects. • Support selection of an OSI-approved license. • Provide the ability for companies to collaborate on projects in a neutral environment. • Act as an agent for the project, receive funds from sponsoring companies, handle trademarks, provide infrastructure as necessary, support with developer relationships, industry and technical events, driving awareness, etc.
Governance	Understand and participate in the project's open governance processes and be an advocate for it.	When establishing new open source projects with industry partners, aim for a balanced governance that gives equal footing to all participants – a governance that welcomes contributors and supports a diverse community.	• Advise hosted projects on best open source governance models. • Help projects to implement their governance.
Access	Foster the culture of free and equal access for everyone.		
Development	• Follow processes. • Recommend improvements.	• Support new projects in creating a number of processes before they launch. These will change over time but it is a huge benefit to have something in place when projects kick off. • Recommend projects document their processes.	

	Open Source Developer	Open source Leadership	Open Source Foundation
Community Structure	• Support the right structure for the size of their project. • Recommend improvements based on their own experience participating in the project.	• Set the project governance and structure with growth and scale in mind. • Adopt practices that worked well in other projects. • Build in the ability to change as the project evolves.	
Releases	• Follow the release cadence when committing to deliver code for a given release. • Evangelize the importance of rhythmic releases. • Provide documentation for their contributions to support good release documentation.	• Promote a given release cadence. • Promote the need for release documentation. • Promote the need for a stable release. • Promote experimentation until the project figures out the right cadence and speed.	
Architecture	Design and implement with scale and growth in mind.	Promote a flexible and modular architecture.	
Communication tools	• Avoid private discussions. • Avoid participating in a closed communication medium (ML, IRC, etc.). • Be inclusive in your communication.	• Ensure that all newly launched or hosted projects offer communication tools used by typical open source projects and are platform agnostic. • Tools are available for anyone to use them and have access to all of the project's communication.	
Transparency	• The project has criteria to promote developers to key positions. • The project has a process that leads to making decisions. • The project has a process to accept incoming code from known entities. • Open communication channels. • Clear governance model.		

	Open Source Developer	Open source Leadership	Open Source Foundation
Development tools	• Use and promote the best open source tools available to support the project's development. • Mentor newcomers into the project on the use of the development tools adopted by the project.	• For any new open source projects your company creates, reply on open source development tools that are accessible to everyone.	• Ensure that all hosted project rely on development tools that are free and available to everyone.
Documentation	• Document your code. • Contribute documentation explaining architectural decisions, code structure, specific modules or features you have implemented, etc. • Review documentation contributed by others; provide feedback and ideas to improve on them. • Provide good headers within source code file. • Respect the project's coding practices and guidelines.	• Prioritize documentation as a parallel track to source code development. • Incentivize developers to provide documentation. • Sponsor interns or technical writers to create documentation for open source projects. • Ensure proper documentation that offer licensing and copyright information.	

Chapter 5

CLOSING

The open source methodology has proved itself over the past several decades that it is better to create software through collaboration and a transparent development process. Open source projects and initiatives provide companies with proven, successful models to collaborate with other companies, create new technologies, and support the development of new communities. Companies across many industries are creating Open Source Program Offices and staffing them with highly skilled individuals to help them drive open source software leadership and gain a critical footprint in this external R&D ecosystem. However, not all open source projects are equally open.

In this paper, we attempt to lay out best practices for open source openness and provide various indicators that may help you gauge the openness of an open source project. Some of these openness perspectives are visible from an external perspective and others are experienced more as a participant in the project.

The paper also provided recommendations on best–case openness scenarios for each of these indicators. If you are an open source developer, an open source leader in your organization, or a leader in an open source foundation, you can enable several best practices to ensure increased openness, transparency, diversity and inclusion in open source projects.

We hope this paper becomes a trigger for new conversations in open source projects on how to be more open, more transparent, and more inclusive.

References

Linux Foundation Enterprise Open Source Guides

https://www.linuxfoundation.org/resources/open-source-guides/

Software Package Data Exchange®

https://spdx.org/

The Software Package Data Exchange® (SPDX®) specification is a standard format for communicating the components, licenses and copyrights associated with software packages.

TODO Group

http://todogroup.org/

The TODO Group is a collection of tech companies who collaborate on the policies, practices, and pragmatics of running an open source program office. Their collaboration is managed as a community project under the Linux Foundation, and they are a resource to companies who are just starting to get their open source programs established.

CHAOSS Project

https://chaoss.community/

The Community Health Analytics Open Source Software project (CHAOSS) is a new Linux Foundation project focused on creating the analytics and metrics to help define community health. The project aims to establish standard implementation-agnostic metrics for measuring community activity, contributions, and health, which are objective and repeatable, and to produce integrated open source software for analyzing software community development.

ACKNOWLEDGMENTS

The author would like to express his sincere appreciation to Jessica Wilkerson (Director of Cybersecurity Research at the Linux Foundation), Christian Paterson (Head of Open Source Governance at Orange), Nithya Ruff (Head of Open Source Practice at Comcast) and Brian Warner (Program Director at the Linux Foundation) for their valuable reviews and feedback. The author is also especially grateful for feedback received from Steve Winslow (Director of Strategic Programs at the Linux Foundation) and the CHAOSS project with notable mention to Matt Germonprez, Kevin Lumbard and Georg Link. This paper has benefited immensely from the experiences and contributions of all reviewers.

FEEDBACK

Suggestions for improvement will be appreciated. Please send **comments** to the author directly.

DISCLAIMER

The opinions expressed in this paper are solely the author's and do not necessarily represent the views of current or past employers. The author would like to apologize in advance for any error or omission and is open for feedback and updates.

ABOUT THE AUTHOR

Ibrahim Haddad (Ph.D.) is the Executive Director of the **LF AI Foundation** that supports and sustains open source innovation in artificial intelligence, machine learning, and deep learning. He previously served as Vice President of R&D and Head of the Open Source Division at Samsung Electronics. At Samsung, he established the global open source division, set and executed Samsung's open source strategy, launched internal and external R&D collaboration projects, supported M&A and corporate VC activities, and represented Samsung in various foundations and consortia. Throughout his career, Haddad held several technology roles at Ericsson Research, the Open Source Development Lab, Motorola, Palm, Hewlett-Packard, and the Linux Foundation. He graduated with Honors from Concordia University (Montréal, Canada) with a Ph.D. in Computer Science, where he was awarded the J. W. McConnell Memorial Graduate Fellowship and the Concordia University 25th Anniversary Fellowship.

Twitter: **@IbrahimAtLinux**
Web: **IbrahimAtLinux.com**
LinkedIn: **linkedin.com/in/ibrahimhaddad**