SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

Open Source, a Mindset and How it Has Transformed Common Platform Enumeration Stack Software Development using Reference Design Kit

A Technical Paper prepared for SCTE by

Khem Raj
Fellow
Comcast
1050 Enterprise Way, Sunnyvale, CA 94089
(408) 768-2010
Khem_raj@comcast.com

© 2022, SCTE®, CableLabs® and, NCTA. All rights reserved.

Table of Contents

Title **Page Number**

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

1. Introduction

"Software is eating the world" quipped Mark Andreessen in 2011 [4]. Today it is apt to say that **"Open Source is eating the software world."** From toasters to a helicopter on Mars, systems are using a robust amount of open source software in their stack. According to the 2021 Open Source Security and Risk Analysis Report by Synopsys [1], 75% of all codebases were comprised of open source code alone.

However, it is still not always easy to work with open source projects 30 years after open source came into being. Organizations and individuals experience challenges in gauging the health of a project: is it stable, secure, sustainable, and will it be there if I depend on it, are common concerns. While the acquisition of open source is free, there is significant effort needed to comply with the hundreds of open source licenses used. To be successful, it is important for organizations that consume open source to contribute back, and to engage with the project. Projects come in all shapes and sizes, with different processes, tools, and funding models; knowing how to navigate open source is a learned skill. As open source companies mature, they also need to learn to release code that they have created, and learn to build communities of users and contributors around it.

2. Open Source Development

Adopting open source development models is flourishing as more businesses discover the advantage of open source projects. Those advantages over proprietary solutions, and the mindset around open source, continue to change as it becomes a more prevalent solution pool. Open source is a mindset which can be applied to solve different problems; it is effectively thriving in software development, but we are seeing an open source development model being used for new hardware development as well, such as Reduced Instruction Set Architecture (RISCV), which involves open source being adopted across the central processing unit (CPU) industry. The recent COVID-19 pandemic also saw open source approaches used for contact tracing and collaboration across the medical industry. While a traditional approach followed a series of steps including requirement, design, implementation, testing and quality assurance (QA), system integration, and deployment all performed serially, in open source the work is done in smaller increments and a "release early release often" approach is adopted. Many of the steps described above are performed in parallel and are iterated more often which allows end users to access the code early in development cycles. This approach is quite successful as it allows faster development and testing, transparency and openness, rapid innovation, enhanced collaboration, and high-quality peer reviewed code.

2.1. Community

All successful open source projects build upon a healthy, growing community of users, developers, maintainers, and promoters. They all share a common goal which is served by the project. The communities are global in nature, comprised of people from diverse backgrounds, which empowers the project with new ways of solving problems with novel ideas. This helps project members to develop new features and capabilities at a faster rate than isolated internal teams could. A collective approach of the community consisting of talented people across the

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

world can deliver solutions faster and more rapidly innovate. This approach establishes an environment of efficient contributions and community team collaboration. An open source community has a strong desire to engage in meaningful productive work, aspiring to develop a sense of belonging, and make it easy for new members to feel at home and connected with others. Open source communities differ in many aspects and each has a unique culture. It cannot be assumed that what is learned as part of one community will be directly applicable in another community. Openness in community fosters authenticity, and members tend to respect it. Therefore, most thriving development communities are those that conduct their activities in open.

2.2. Reliability and Security

Open source projects host the source code in open repository infrastructures such as GitHub or GitLab, making the code more accessible to a wider audience of developers and individuals. This approach results in more individuals looking into the source code regularly. Some users will test newly developed code and provide feedback to developers. Because the code is reviewed in an open environment, healthy reviews are the norm, making the code more robust prior to acceptance[7]. Since code is subjected to much more thorough review and testing, it also tends to be more secure [6], and any security issues that do arise afterwards are handled briskly. Updates are frequent which provides a smaller window for exploiting any weaknesses in code. A major motivation in decision-making is to select the best possible solution. In some cases, this means selecting best technical solution instead of being influenced by other factors which might provide a quick but technically inferior solution. This merit-based approach is key reason for open source projects being more reliable and secure[5]. It is good to use security response time for patching security related bugs of a given project as part of selection criteria for an open source project.

2.3. Transparency

In an open-source community, code is available openly, and all discussions happen on open channels such as internet relay chat (IRC), web forums, or through mailing lists. All decisions regarding new features and defects also follow open channels. This information is very valuable for users as they can see the technical merits or limitations of the solution and avoid any unwanted surprises later in deployments. Open channels also encourage users to discuss the code issues they encounter with the community and forge collaborative solutions. This also avoids vendor lock-in which might be a consequence of using closed solutions.

2.4. Support and Maintainance

Any software will need support, and open source projects are no exception. While proprietary solutions are extensively supported by vendors providing a single point of contact, this model may not translate directly to open source, as the software is created by a number of developers covering different parts of the project. Therefore, it is important to look into the support aspect of projects and find third-party dedicated support or use the online community-provided support if that is sufficient. There may be no upfront cost to using an open source project, though there could be additional cost due to integration or maintenance of the project needed in the business solution. After some time, an open source project may stop supporting a prior version that was deployed in products, which could mean additional cost to engage third-party software support. It is beneficial to understand these hidden costs beforehand for better estimation of support costs

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

for the products. It is worth stating that the costs for support and maintenance will be reduced compared to a traditional approach, but it will not be zero, as some might proclaim or assume since the code is available for free.

2.5. Working with Open Source Communities

Each open source community is unique and has subtle characteristics which need to be well understood for one to effectively participate in the community. It is important to understand that communities are united by a common interest; finding this interest and understanding the goals of a given project is of utmost importance. Communities have laid out certain rules of engagement, though not all of them may be documented. These can be learned by reading through community channels, forums, and mailing lists, or by interacting with experienced community members.

Usually, projects have a README file which is a good place to get acquainted with the project. There are other common files which may be provided by projects including:

- CONTRIBUTING file, which would list contributing procedures and tools required.
- MAINTAINERS file, which is a good resource to learn about key developers and maintainers who can be valuable points of contact as you start engaging with the community.
- LICENSE or COPYING file, which lists the open source license governing the content of the source provided by the project. This is crucial information which should be well understood. It may require review by your organization's legal team to provide guidance on the mode of engagement that can be pursued.

Some projects with a large codebase and community create governing bodies to take care of project administration and future technical directions. These governing bodies are instrumental in shepherding the project and it is important to understand the role and functions of such committees if they exist.

There are frequently multiple projects available for a given requirement in a product. Therefore, it becomes important to select the best fit. The maturity of a project is an important mark; looking at the number of commits and rate of commits can give a good hint of project maturity. Some infrastructures such as GitHub also provides stars for a project which are awarded by users of the project; more stars means that more users found the project useful for their needs. This could be another parameter to check. Regular release cycles can be used to deduce how actively code is maintained. If a project does not have a deterministic release cadence it becomes difficult to engage with such projects because it would be hard to align with your project roadmap. In other instances, the number of open defects and the rate at which the issues are handled can serve as a good mark of project development. A larger number of contributors is a good sign as well, since it would mean that it would be easier to find help when needed, while diversity in committers could also be indication of good quality within a project. There are other markers that can indicate the strength of a project including Continuous Integration (CI), good documentation, and examples and lists of other projects using or depending upon the project. Open source project space is dynamic, and projects can be started and soon become stale or dead. The risk that arises due to this unpredictability can be minimized by taking precautions as mentioned above.

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

Contributing to a project is important and is the lifeline of any open source project. If you depend on an open source project, you should consider contributing to the project. Doing so would benefit your use case as it also supports the overall project. When we speak of contributing, it is often equated with contribution made in the form of code. Even though it is a significant contribution category, this is not the only way to contribute to a project. There are other aspects of projects that can benefit from contributions as well. Documentation is one key aspect of any project and does not strictly involve code right away. Open source projects interact via online forums or mailing lists, where many questions are asked, so active members can contribute by answering these questions. Website designs, infrastructure setups, providing computers for CI, defect triaging, organizing events, connecting community members, promoting and marketing the project, and mentoring new community members are additional ways to contribute to a project in a significant manner.

As one becomes a regular contributor, they can begin providing inputs for project directions along with new features and design decisions. These are deeper engagements which could require a significant time investment. Doing this, however, would be impactful and help steer the project and promote you into an influencer or leadership role in a project or community.

2.6. Open Source Licensing

Most open source software projects have licenses. If there is no license mentioned explicitly, then you should assume the software is not being licensed and therefore may not be copied, distributed, modified or otherwise used.

Some of the common open source licenses are GPL family, Apache 2.0, MIT, Mozilla Public license, and BSD family. Broadly, these licenses are classified as either copyleft or permissive. Copyleft licenses typically demand reciprocity and require that any derivative work is also licensed under the same obligations. The GPL v3 license is an example of a copyleft license. Some licenses are permissive, which grants license to use and re-license. BSD, MIT, and Apache are common examples of permissive licenses. Proprietary licenses are more restrictive than open source licenses on granting rights; these are typically non-free licenses. It is very important to learn and understand the implications of the license of a specific open source project. It will help in understanding the contribution and distribution mechanisms of the source code and its derivatives.

There are obligations to fulfill as a consumer of the project; a different set of obligations may be required to be fulfilled when distributing the project to your end-users and consumers. Generally, Open Source Program Office (OSPO) departments which help organizations in formulating the open source policies are the best resources to provide a detailed understanding of the open source licenses to developers and technical teams.

When open source components are used as part of a product, there may be more than one license which is in play. Therefore, it is absolutely important to be aware of every open source license used in products and create a license manifest. There are commercial and open source tools which can come to aid in building the open source license manifests. It is important to note that

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

these tools are not the final authority on signing off these manifests; assigned auditors should review the licenses and ensure that they are listed as required. These manifests are living documents, which can change as soon as the component list or dependencies change. As such, it is beneficial to generate these manifests along with production builds and have these manifests correspond to each exact source that is being used in each release.

For scalability, it is good to define an intake process in which teams are provided with the correct information and guidelines to choose a component and assess its license. This approach helps in making appropriate decisions early and avoids license-related showstoppers at later stages of a project, which can result in lost time, and in many cases has caused projects to be cancelled. A vetted process to get approval for using a particular license is good to have, with checks and balances to ensure that these kinds of problems are avoided. It is good practice to require that developers regularly take license-related training to stay current on open source licensing trends.

When creating open source projects and seeding them, it is also important to license the code appropriately. There could be many factors influencing this choice, including dependencies of the project, perhaps the inclusion of another set of open source components, or the situation in which code is licensed under a license which may be incompatible with the license you are trying to use for the new project. It is also important to align the license with project goals and community involvement. Some licenses may require more effort to fulfill compliance within the conditions in which the project will be used. Therefore, software architecture including static or shared linking may also play a part in determining which licenses are more suitable than others. It is important to consult with knowledgeable teams, especially the legal team supporting your group, to come to a decision on choosing a license. Choosing the "wrong" license can have serious consequences, so it is important to choose the best license for each project.

2.7. Supporting Key Open Source Projects

Open source dependencies for products can run deep, and in many cases the product may have a critical dependency on a potentially dead open source project. A typical response is to keep using dead code, as switching to an alternative could be quite disruptive if it would come at a time which was not planned. It is important to find your open source dependencies and evaluate projects closely for their health and long-term viability. Doing so provides a good overview of the strength of a product's foundation. An open source project may be in need of computers for hosting testing infrastructure, there may be a downward trend in contributors, or the project may be maintained by a single person in their free time, making it important to find out how you may be called on to help a project. You may be able to support a project by donating funds, contributing bug fixes, making improvements, or introducing features. Some projects require conference sponsorship or event hosting. Creating a comprehensive support plan can be useful in maintaining continuity and helps avoid surprises when something in the community breaks down.

2.8. Seeding New Open Source Communities

Open source engagement evolves from being a user to becoming a participant, then further to being a contributor, influencer, or maintainer. These are helpful experiences to gain before starting a new community or seeding a project. Starting a new open source project involves many

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

activities besides making code available. There are millions of open source projects, and it is likely that a similar implementation is already available; how you differentiate your project is a key aspect to consider. An open source project would not go far without a vibrant and healthy community around it. When starting a new project, it is a big challenge to build a community of users, developers, and promoters. As such, one needs to define clear goals and a meaningful value for the audience that will drive its consumption. Doing so will help with community growth; there must be clear directions for someone new to get started with the project, and contribution guidelines and steps should be crisply defined. As communities are comprised of humans, it is important that members see a clear, tangible value in being part of community. Communities develop culture. It is not something that will be pre-defined; and, as it is essential for a community to endure, the proper care and feeding of community is necessary. Engagement models will define how new members are on-boarded, how friction is removed for regular members, and how core contributors are incentivized. This documentation will provide a good process to facilitate learning for members.

2.9. Open Source Program Office (OSPO)

As open source adoption has become prevalent, scaling is a challenge, particularly in large organizations. Some teams may be open source savvy and forge an open source-based solution which could be quite optimized and best in class, though it is not easy to replicate that level of success across different products and teams without effort. OSPOs are established to tackle such problems. OSPOs act as an interface between open source communities and the organization. OSPOs also helps connect various internal teams wanting to benefit from open source. They can be instrumental in defining open source policies for the company. Open source communities are quite dynamic in nature, and it is often required to have someone to actively look into the ongoing developments and make meaning of them from an organization's point of view. As noted above, projects periodically change their licensing terms. If these projects are used in a company's products, the OSPO can help in interpreting these changes and monitor relevant changes. OSPOs help in optimizing open source consumption, collaboration, and contributions. They keep track of statistics and data and monitor the progress on a regular basis. This type of data is important in measuring productivity relative to open source goals that the organization might have set forth. There are various events and conferences related to different open source technologies, and as projects are organized across different geographies, OSPOs track event calendars and bring vital information to the company teams. The OSPO also connects internal team members who may be participating in the same events. They can also help with internal initiatives to drive the open source mindset inside the organization, to promote inter-team collaboration and code sharing, resulting in similar benefits for proprietary code development as well. To assist with career growth in engineering, developers might be interested in speaking opportunities at open source events. While the call for presentations for these events happens months ahead of the event itself, the OSPO keeps this information current and keeps the teams informed about the deadlines. OSPOs can also help in adopting various processes for open source compliance and recommending scalable tools across the organization. Given these important functions, OSPOs are becoming an integral part of modern organizations.

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

2.10. Attracting Best Talent

Due to their open nature, ease of access to source code, and other related tools and infrastructure, open source projects provide fertile ground for training new developers. Many developers are enthusiastic about working in an open source project or environment where an "open first" approach is used for software development. If an organization is involved in using, collaborating, and contributing to open source projects, it can be a big incentive for top class developers to seek positions there. If an organization employs well-known open source developers, it motivates other developers to join the company to work and learn from the best developers and collaborators. To attract talented developers to your organization, showcasing your community participation, involvement, and contribution can go a long way. Doing this creates a large pool of developers and engineers with experience in open source technology, which in turn ensures that there are enough people available to work on future critical open source projects.

3. RDK and Open Source

Reference Design Kit (RDK) provides a set of software components which are used to build software stacks for video clients, video gateways, broadband routers and gateways, smart camera solutions, wireless access points, and more devices used in the home. RDK is used to quickly build efficient stacks. RDK has transformed cable software stacks by providing a robust platform backed by open source solutions and technologies. The initial profiles were video client and gateway devices, which soon expanded into broadband profiles, building routers, access points, and providing Data Over Cable Service Interface Specification (DOCSIS®), digital subscriber line (DSL), and fiber wide area network (WAN) based devices. In the future we will see 5G modems and routers, as well as smart TVs being built using the RDK software platform.

RDK started as a foundation with RDK Management being the custodian of the code and infrastructure. RDK has adopted open source best practices and improved upon its own processes iteratively over time, establishing clear contribution guidelines and processes which are readily available for community. There are also webinars and How-To documents made available for new and seasoned contributors. The community growth is measured in terms of new members and contributors joining the project. It also measures lines of code growth and number of changesets submitted and accepted on a quarterly basis. These measurements offer key insights into community engagement and growth. RDK started with most of the code being under RDK license but has since migrated to an Apache 2.0 open source license for all new code and also relicensed most of the original code under Apache 2.0. All new components are released under an Apache 2.0 license and made open source from day one. In some cases, components are being incepted on GitHub and developed directly in the open with community collaboration.

Such best practices have been adopted and tuned over a period of time as the project grew its userbase and the needs of the community for open collaboration grew beyond the initial set of members.

RDK has also regularly hosted community events including the annual RDK Conference and RDK Tech Summit. Attendance and topics discussed in these events have grown multifold. Where in the first conference topics focused mainly on video projects, topics now include broadband technologies, wireless connectivity, augmented reality, smart speakers, smart

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

cameras, and new open source code and features that are being created by the RDK community. These events also serve as key collaboration and networking opportunities for the community of developers and users. There are additional events like coding challenges which are organized throughout the year.

RDK has also embraced popular open source platforms for ease of development and to broaden community outreach. Projects are mirrored on GitHub as well. Starting with just a few projects, RDK today contributes directly to many upstream projects including Yocto, Webkit, and Gstreamer. At the same time, there are RDK initiated projects which are also being used beyond RDK in other projects. This fact demonstrates how RDK started as an open source user/consumer and slowly graduated to a participant, contributing to key projects, before finally producing and seeding projects and growing the community. RDK's commitment to open source has led to leadership and influence in the cable industry in particular. RDK now has templates for open sourcing projects. These templates include key files that an open source project should have. Members can utilize these templates to quickly learn and become proficient in open source processes. It now also helps new members to become good at using RDK components by offering trainings, webinars, and dedicated community support. There is a clear path for members to become contributors and maintainers of components.

As contributions grow, conflicts will happen where a decision to adopt a given implementation or another must be made. RDK has established Technical Advisory Boards for different profiles. These committees of maintainers and contributors broker a decision based on the technical merits of contributions and the RDK roadmap.

The RDK open source community also keeps a keen eye on the latest trends and developments in other open source communities and adopts tools or best practices that were successful for other communities such as adopting Slack for instant messaging and collaboration.

From custom licensing requirements to an "open first" development mindset, RDK has transformed itself and keeps learning from open source development and processes, incorporating these learnings to address its own community challenges.

3.1. RDK Infrastructure

RDK is hosted on https://rdkcentral.com, which includes instructions on how to join the community, documentation, and the code itself. RDK uses self-hosted gerrit and GitHub cloud infrastructure to serve the community. Jira is used for workflow management, defect tracking, and technical support. There is extensive wiki documentation available for technical know-how, webinars, FAQs, and more. Additionally, regular meetings are held for roadmap sharing, and general community synchronization and updates. Special interest groups (SIGs) are formed as needed to forge new features or address large scale problems such as security vulnerabilities.

3.2. RDK development and workflow

All new components for RDK are developed in open source as community members and developers contribute code on a regular basis. Contributions go through an established intake process. This process involves scanning code for license compliance, build verification, runtime

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

sanity testing, and code review. In addition, some members also validate the patches in their internal setup and provide the testing results back as community contributions. This approach establishes a robust intake process which provides sufficient testing during code development and helps avoid regressions that may be introduced by newly added code.

3.3. RDK Collaboration

The RDK ecosystem consists of different types of members. Hardware manufacturers, typically consisting of original equipment manufacturers (OEMs), silicon on chip (SOC) providers, and specialized hardware add-ons, build hardware for specific products. Independent software vendors provide value on top of RDK platforms. Application developers build and port their applications to the platform. Service providers use these products as vehicles to deliver the services to customers. Students and universities work with the RDK platform on experimenting with technologies, learning, and research. As we can see, there is a wide array of users and contributors. Each could be working on enhancing a given segment of the RDK, such as a SOC vendor interested in unleashing the power of hardware capabilities of CPUs and other co-processing units. They would require working toolchains and infrastructure to build upon. Similarly, application developers would require a working stack on production hardware or reference systems. RDK enables these key elements by providing infrastructure, tools, and techniques upon which everyone can build. A common set of workflows for code development and acceptance criteria means that contributors are adding value for each other, and an established code quality is maintained. A common set of references are maintained as hardware platforms which are easily accessible to the community and are close enough to product hardware which users might be working on. This result provides a quick on-ramp for porting and increasing feature velocity.

3.4. Key Takeaways from Open Collaboration

The open platform that RDK has built provides many additional benefits. It has improved code reuse; since code is openly available, there is less duplication. A new stack uses common building blocks which are the same across all profiles, and scale horizontally. This stack also includes common build infrastructure and CI systems. It has resulted in better use of resources and enabled the community to avoid redundant work. Promoting re-use means that a modular software design has emerged. A departure from a monolithic mindset to a component-based mindset has been an on-going transformation. As components take shape, it has improved code ownership and maintenance, as expert teams are able to function independently while developing components maintained by them. It is an engineering community with collaboration mainly around source code. This construct has provided opportunities for any new developer, student, or other entity to submit pull requests and contribute. The feedback loop is essential for learning, and it has helped individual members and member organizations develop in-house expertise rapidly. Open source RDK has also helped training organizations to develop good programs and they can easily keep these modules up to date as the code changes, providing fresh courses and training events. These programs serve as a critical part in growing the developer base. Application developers can build their applications once and deploy many times on different RDK-based systems, streamlining the efforts to improve the application experience. There is a rapid expansion of device profiles which can be attributed to the key points discussed above.

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

4. Conclusion

The use of open source development methods and processes is on the rise, offering significant advantages over traditional closed software development methods. Major pieces of many complex software stacks today consist of open source software, and its share is on the rise as well. It is important to become a wise consumer and thoughtful producer of open source software. Doing this requires a deep understanding of open source communities, open source licenses, and various projects involved in building products. Strategically aiding and sustaining the communities supporting and building critical software is essential. Scaling to use open source software across larger organizations is a challenge which is being addressed by dedicated Open Source Program Offices which serve as liaisons between internal development teams and reach out to open source communities. The "open first" mindset is also on the rise, where open source solutions are sought first and developed and customized.

Abbreviations

CPU	central processing unit
DOCSIS	Data Over Cable Service Interface Specification
DSL	digital subscriber line
IRC	internet relay chat
OEM	original equipment manufacturer
OSPO	open source program office
QA	quality assurance
RISCV	reduced instruction set computer five
RDK	reference design kit
SIG	special interest group
SOC	silicon on chip
WAN	wide area network

Bibliography & References

[1] 2022 Open Source security and risk analysis report - https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/rep-ossra-2022.pd

[2] People Powered – How communities can supercharge your business, brand and teams – Jono Bacon

[3] 4 Stages of Open Source Consciousness – Khem Raj

[4] Why Software Is Eating the World – https://a16z.com/2011/08/20/why-software-is-eating-the-world/

[5] Reliability Issues in Open Source Software - https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.434&rep=rep1&type=pdf

[6] Is Open Source Software More Secure than proprietary Products - https://www.govtech.com/security/is-open-source-software-more-secure.html

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.

[7] Open Source Case for Business - https://opensource.org/advocacy/case_for_business.php

© 2022, SCTE® CableLabs® and NCTA. All rights reserved.