FINOS

Fintech
Open Source
Foundation

Business Value of Open Source for Financial Services Firms

Produced in collaboration with

redhat | wipro

Table of Contents

Contributors

This whitepaper was produced collaboratively by FINOS, Red Hat, and Wipro. We're grateful to our partners for working with us to empower the financial services industry to get the most out of open source collaboration.

Red Hat is the world's leading provider of open source software solutions, using a community-powered approach to reliable and high-performing cloud, Linux, middleware, storage, and virtualization technologies. Red Hat also offers award-winning support, training, and consulting services. As a connective hub in a global network of enterprises, partners, and open source communities, Red Hat helps create relevant, innovative technologies that liberate resources for growth and prepare customers for the future of IT.

Wipro Limited (NYSE: WIT, BSE: 507685, NSE: WIPRO) is a leading global information technology, consulting, and business process services company. We harness the power of cognitive computing, hyper-automation, robotics, cloud, analytics, and emerging technologies to help our clients adapt to the digital world and make them successful. A company recognized globally for its comprehensive portfolio of services, strong commitment to sustainability, and good corporate citizenship, we have over 175,000 dedicated employees serving clients across six continents. Together, we discover ideas and connect the dots to build a better and a bold new future.

The Fintech Open Source Foundation (FINOS) is a nonprofit foundation promoting innovation and interoperability in financial technology through industry-wide collaboration on open source software and open standards. FINOS's community includes financial services firms, financial technology consumers, service, and solution providers, and other constituents. The Foundation provides a trusted, neutral forum for community collaboration, offering an efficient, compliant open source collaboration infrastructure and transparent, community-driven technical governance.

Authors

Andrew Aitken

GM & Global Open Source Practice Leader,
Wipro

Andrew Aitken is the GM & Global Open Source Practice leader at Wipro. His focus is driving open source adoption for enterprises through differentiated open source services, solutions, strategic partnerships, and ecosystem collaborations. He and his team of open source experts work with customers to deliver business value through open source adoption, to drive business agility, and foster innovation.

He has served as an open source expert to the White House and California Senate and as a guest lecturer for Stanford's Entrepreneur program. He is an investor in Mautic and on the Board of Advisors of various open source companies. He is a past Board Member of OSEHRA, an initiative spearheaded by the U.S. Department of Veterans Affairs to provide cost effective healthcare solutions for veterans through the development of an open source electronic health records community. He has personally worked with organizations such as IBM, Capital One, Thomson Reuters, the U.S. Navy, Microsoft, the government of Japan, and dozens of startups, assisting them with developing their open source strategies.

Mr. Aitken has been a pioneer in the development of governance and commercialization models for open source. He is considered a thought leader in the emerging trend of applying open source development methods to the corporate development and innovation processes, known as Inner-source.

Leslie Hawthorn

Sr. Principal Technical Program Manager,
Open Source and Standards Team, Office of the CTO, Red Hat

Leslie Hawthorn leads the Open Source Executive Practice at Red Hat, helping customers learn how to most effectively use open source software, and to build out their processes and policies.

Leslie started her career in open source with Google's Open Source Programs Office where her first responsibility was turning the Google Summer of Code into a recurring, annual program and expanding it into other areas of contribution such as Google Code-In. She currently serves the FINOS Foundation as a contributor to the Open Source Readiness Working Group and is Board Emeritus at the Open Source Initiative.

Aaron Williamson

General Counsel and Director of Governance,
Fintech Open Source Foundation (FINOS)

In addition to managing the Foundation's legal affairs, Aaron leads the FINOS Open Source Readiness Program, helping members to develop policies and processes that enable productive engagement with open source. He also co-organizes the FINOS Open Source Strategy Forum, an annual conference on open source in financial services. Previously, Aaron has counseled both commercial software producers and nonprofit foundations on legal issues related to open source software, both in private practice and at other nonprofit organizations.

Executive Summary

The financial technology industry is moving into a new era of engagement with open source technologies and communities. Having used open source components for decades to build their products and run their datacenters, financial services and fintech firms are increasingly realizing that active participation in open source development is necessary to fully realize their goals to spur innovation, reduce development costs, and attract top talent. This paper outlines how.

Consuming Open Source Software

- Use of open source components in software development, whether for internal or external consumption, is essential to drive your product to market on a reasonable timetable.
- Make it easy for developers to request and receive approval to use new open source components in software development.
- Evaluate licenses case-by-case. A license that's not appropriate for a product delivered to customers might be perfectly acceptable for internal products.
- Develop an open source strategy that preserves your developer resources for high-value projects by encouraging open source use and contribution everywhere else.

Managing Risk Related to Open Source Software Consumption

- Understand the different types of open source projects and communities in existence and build institutional knowledge regarding their differing risk vectors.
- Support for open source software is not one-size-fits-all. Design your requirements to fit your products, the open source components and applications you're using, and your risk model.
- Automate checks for security issues, license compliance, and data and IP leakage. Work these into your software development lifecycle as seamlessly as possible.

Contributing to Open Source Software

- Contributing your improvements to the open source projects you rely on is not altruistic. It makes it easier to integrate future versions and respond to security threats, enables community enhancements, supports the project's health, and makes your developers happy.
- Contributions to third-party projects come in all sizes, from bug fixes to valuable new features. Design a review and approval process that weighs each contribution appropriately.

Publishing Internal Projects as Open Source Software

- "Open-sourcing" your solution to a common problem can help establish a de facto standard, bringing the rest of the industry to your way of working and saving you effort.
- An open source solution can also commoditize the relevant market, pushing the competitive frontier closer to where you provide value.
- Open source activity bolsters your company's reputation in the open source community and among potential recruits. Increasingly, developers want to work for companies that understand and participate in open source development.
- Strategic open source engagement means understanding where your value lies. Don't be hamstrung by assumptions that all IP is more valuable if kept proprietary.

Introduction

Open source: it's not if or when but how much and how fast

"We want to operate more like a technology company providing financial services than a financial services company using technology to deliver our services, and open source will be our transformation catalyst."

This is a common sentiment expressed by financial services executives across the globe, from insurance to wealth management to corporate and retail banking. Open source has moved well beyond the status of simple technologies for gamers or hobbyists and into the executive suite as a strategic tool to drive enterprise transformation. The conversation is no longer focused solely on saving money, or on concerns about scalability, security, and supportability, but about significantly improved innovation processes, greatly reduced time-to-market, brand management, improved customer engagement, and the ability to attract and retain one of today's most in demand resources—developers.

In the offices of the most senior IT executives in one of the most conservative and highly regulated industries, you'll hear conversations like; "We have to work with the community to decrease the time it takes them to accept our patch", or "How many pull requests did we get on GitHub last week?".

Despite open source underpinning every major technology evolution today and everyone saying; "We're going to be the Facebook or Netflix or Uber of our industry" (all built on open source), there are practical roadblocks and misconceptions that stand in the way. These barriers hold financial services companies back from realizing many of the transformative benefits of open source.

Open source can be hard to integrate and in many technology segments the pace of its evolution and adoption of a technology sometimes outpaces the maturity of its ecosystem. There may be significant resource and governance constraints on the project, or limited attention to compliance. Best practices for support and engagement models are still evolving.

Further, as financial services organizations have begun to open source their own software over the last few years, there have been failures, or at least irrational expectations that have not been met and so considered failures. In many instances, these are due to lack of adequate funding, budget cycles being misaligned with the length of time required to build a viable community, or projects being started for personal reasons rather than sound strategic objectives.

Simply put, it is very hard to build a vibrant, self-sustaining community. It requires absolute commitment over an extended period, the realization that to grow, you must share control, and understanding that developers care about behavior and actions (not what you say). Finally, everything must be done in an open and transparent manner— characteristics often not found in the highly regimented and regulated financial services industry.

It's common to think of open source as some mix of getting free stuff (consuming third-party open source software) and "contributing to the community" (contributing to or publishing in-house software as open source). These rough characterizations disguise both the responsibilities that come with consuming open source and the very real benefits that come with participating actively in open source development.

Consuming Open Source Software

The most obvious value companies get from open source software is by consuming it. By "consuming" open source, we mean:

- incorporating open source libraries and components into custom applications;
- using open source applications internally on servers, personal computers, and appliances like routers; and
- deploying cloud native applications and migrating workloads to public clouds that are built upon open source infrastructure and analytics technologies.

The perceived primary benefit of running open source software is straightforward: if you can get the job done with freely available open source software that you can extend as you see fit, you can save software licensing costs and consulting fees. Below, we'll focus on more of the benefits achievable through consuming open source in your own software development efforts.

Benefits of building on open source

Today, even the simplest software is merely the tip of a vast iceberg of code. A basic website may use a web application framework (like AngularJS or Django) for server-side functionality and a user interface utility library (like Bootstrap) to provide visual consistency and basic interactivity. These tools provide essential front- and back-end functionality that every website needs, as well as advanced features for more complex sites. They each comprise tens of thousands of lines of code, the product of thousands of hours of coding, revising, refactoring, and testing. And they're all open source.

Your developers could write their own tools for these tasks or you could license similar tools from a commercial vendor. But these existing open source tools are powerful, mature, well-documented, and each maintained by a thriving community of corporate and independent contributors. To varying degrees, the story is the same for every application your developers need to build: there's a vast selection of open source tools—from the trivial to the essential—to help them get the job done. By leveraging these resources, your developers and your company will realize substantial benefits.

Reduced Time-to-Market

As it may be clear from the example of the simple web app above, using open source components for under-the-hood functionality can dramatically reduce the development time and resources necessary to build an application. But today, open source is so widespread and so essential to modern software development, it's perhaps more important to consider the inverse: without depending heavily on open source components, you'll never make it to market.

Without depending heavily on open source components, you'll never make it to market.

Your competition is looking for every opportunity to beat you on features, cost, and development time, and if they're taking advantage of open source and you're not, they'll do it every time. The question is not whether to use open source but how to do it more strategically, efficiently, and extensively than your competitors.

Although financial services have long been adopters of open source in the infrastructure space they are still in the earlier stages of adoption for applications and open source collaboration. Many companies continue to face challenges understanding the implications of open source licensing, IP risk, and support. The companies that move quickly to come to terms with the manageable risks associated with open source software will put a lot of daylight between themselves and those that insist on a conservative approach.

CASE STUDY

One European-based global financial services firm is deploying an enterprise-wide open source monitoring and analytics solution based on five open source projects in order to reduce licensing costs and most importantly reduce time-to-implementation.

There are several ways that a company already using open source can increase the value it gets from that technology:

- Revise open source review and approval processes to be more responsive to developer requests in order to remove friction and speed development.
- Push approval authority for new open source uses as far down the org chart as practicable. This puts the decision in the hands of the person best-suited to evaluate the request (e.g. the developer's direct manager), reduces delay, and preserves the time of higher-level managers.
- Establish or expand opportunities for developers to contribute to open source. Read below to learn why the greatest benefits of open source are reserved for those who participate in its development.
- Re-evaluate restrictions on open source usage. For example, rather than prohibiting copyleft licenses like the GNU General Public License (GPL), consider permitting use for low-risk use cases like client-side developer tools and server-side applications.

Preserve Resources for High-Value Development

The vast majority of the 1000's of applications running in any bank do not provide a competitive advantage. They are essential systems that span a vast range of functionality across front, middle, and back office, including software for trade feeds, break resolution, anti-money laundering, settlement, data storage, and retrieval, and even facilities management. These are systems every bank must have, but no one is making money from them. They're the bridges and roads that high-value applications—single-dealer platforms, algorithmic trading systems, proprietary research—use to move data from point A to point B.

The less work a company has to put in to building and maintaining this infrastructure, the more resources it has to devote to the products that set it apart from its competition. This is true whether it's using open source components to accelerate the development of a single application, or using open source applications in place of proprietary infrastructure.

Keep Developers Happy and Productive

For an entire generation of software developers entering the workforce, software development is now synonymous with open source. They've begun every software project by assembling open source components — their preferred open source programming language (like JavaScript, Python, Haskell, C#, or Ruby), database (like MySQL, MongoDB, or PostgreSQL), application development framework (like AngularJS, Django, or Rails), and maybe a server-side runtime environment (like Node.js or Twisted) — and building their work on this stable, tested, well-documented foundation.

This is simply how software is built today. An open source policy can ensure that developers have ready access to the essential tools of their trade, or it can put barriers in their way by requiring excessive process or needless approvals. A good policy won't sacrifice productivity on the altar of risk mitigation but will, instead, streamline development in a manner consistent with a firm's policies.

A good policy won't sacrifice productivity on the altar of risk mitigation but will, instead, streamline development in a manner consistent with a firm's policies.

Further, developers are keenly aware of the value of working with open source software for their own professional development. Remaining familiar with open source technologies provides a pathway for continuous learning of their craft, and participation in open source communities builds their own personal brand reputation as developers. Many companies have created employment policies that explicitly allow for contribution to open source projects simply because they are unable to attract or retain the talent they require without one.

Anytime you enhance an open source product to better suit your needs, chances are those enhancements will be useful to another member of the community. Contributing them back to the project gives others the opportunity to use and improve those features—changes your company will benefit from.

Support the Health of Projects You Rely On

Contributions are the food that nourish open source projects. A steady stream of contributions from a robust community of users keeps a project healthy—it makes the maintainers' efforts rewarding, creates a sense of community that encourages others to join in, and it ensures that there are always candidates for project leadership in the event one of the existing leaders needs to step down.

By empowering your developers to contribute to open source projects and participate in the surrounding communities, you help to ensure that the projects you depend upon remain actively maintained and well supported.

Empower, Retain, and Attract Developers

Apart from these benefits to your software's security, functionality, and longevity, empowering your developers to participate in open source communities will improve their quality of life. Developers want to contribute to the open source projects they work with; increasingly, the most sought-after developers demand it.

Just as your developers are more productive when they can rely on readily available open source components, they preserve their time, energy, and enthusiasm by contributing back their modifications to the projects that produce them. The alternative—revisiting their prior work every time the upstream project releases a new version (or worse, failing to update the modified component)—is a recipe for frustration.

By contrast participating in open source projects improves developers' jobs in tangible and intangible ways. Beyond making the developer's job easier, giving back to the projects they benefit from feels good. This is what we mean when we talk about open source "community"—the satisfaction, appreciation, recognition, and relationships that arise from solving shared problems with peers.

Managing potential risks related to open source usage

When a financial services firm first looks to expand its open source engagement, the question of risk comes up a lot. Open source software is not inherently "riskier", nor less secure, than proprietary software, nor are the attendant risks categorically different, but companies do need to approach risk differently in the open source context.

One consistent area of concern is support: where will the company turn to for support when an open source component fails or requires further development? There are many answers, depending on the context and business requirements.

Project Governance: Community, Corporate, and Commercial

"Open source software" is far from a monolith; projects vary widely in how they're developed, who has a say in the development process, and what the motivations of the developers are. Open source projects can be broadly divided into three categories: community projects, corporate/foundation projects, and commercial projects.

A community project is not sponsored or controlled by a single corporation, but is instead maintained by a community of individual contributors. Those individuals may be contributing on behalf of their employers, but none of those companies has fiat control over project decisions. A corporate or foundation-based project is established and run by one or more companies, either independently or as a group within a foundation construct, which typically maintain significant control over the project's roadmap and release schedule. Finally, a commercial open source project is a corporate open source project whose corporate sponsor sells paid licenses or subscriptions for some version of the project software (typically an enhanced "premium" version or simply one with the open source license restrictions removed) or support and services.

When evaluating an open source technology for inclusion in your company's software, it may be worth considering which type of project produced it, particularly if the component is mission-critical. It may be easier to influence the direction of a community project's development, but it may be easier to get reliable paid support from a commercial open source provider. A corporate project may progress more rapidly and devote more resources to fixing security vulnerabilities but the corporate sponsor's development goals may ultimately conflict with your own.

Being aware of the features and bugs inherent in these different models can help your team choose the most appropriate tool for a given requirement.

Open source support models

A concern commonly raised by companies that are new to open source is whether commercial third-party support will be available if the open source component fails or requires modifications. Some have gone as far as to require software development teams to engage a commercial support provider before onboarding any open source component.

This approach can stall both innovation and productivity in development teams looking to move quickly in their development processes. A more balanced approach is to allow developers to consume open source software and ensure that appropriate (commercial, if necessary) support is obtained before moving into production. There are now numerous vendors who will provide support for nearly all 3rd party packages and libraries.

There are now numerous vendors who will provide support for nearly all 3rd party packages and libraries.

Support for open source software comes in several flavors. Some corporate open source providers offer support on a “freemium” model: there’s no difference between the free and paid versions of their open source offering except that the latter comes with a service-level agreement. Others, like Red Hat and NodeSource, provide additional functionality or more rapid updates to their paid support customers, in addition to one-on-one support. Finally, it’s often possible to contract on a time-and-materials basis for support or custom development on specific issues.

For the more popular open source tools, all of these options may be available, either from a corporate sponsor or a third-party. This highlights one of the chief benefits of open source software: avoiding lock-in to a single vendor for ongoing development and support. Even with a commercial open source provider, where the relationship may in all other respects be identical to a typical software vendor relationship, the customer isn’t stranded if the relationship breaks down; they can usually find someone else to provide support or take it on themselves.

What kind of support is available—and how much of the support burden your team will bear—also depends on where and how you’re using the open source software. If the component is incorporated into an internal software product, support may necessarily come in the form of time-and-materials consulting, because issues may arise due to interactions with your software. If you’re running a standalone open source platform or application on your servers, a wider range of support options may be available, but they’ll all require someone within your organization to assist the support provider to some degree. Finally, open source software hosted offsite as a service may require very little of your resources to support, but that benefit may come with significant limitations on customization.

One North American bank has supported a committer on the Apache project for over a decade because they have deemed it a critical technology and want to understand how the project is evolving and have a voice into that process.

Evaluating the “risk” of using an open source tool

For financial services firms who face strict regulatory restrictions that vary widely across geographies and industry segments, ”risk” is the watchword: what is the risk of relying on an open source tool instead of proprietary or commercial software? While open source software may be “free” to license, there may be costs associated with getting the updates, support, and other benefits commonly associated with a commercial software license. The question is, are those costs worth it? Our answer is that, with the right approach, greater reliance on open source software can reduce cost across the enterprise while increasing return on technology investments.

The risk associated with a particular open source component depends on the intended use, the characteristics of the tool itself, and the tool’s provenance. If the component will support some minor functionality in a nonessential application, the stakes are low; you might reasonably rely on an open source library written by a college student who hasn’t issued an update in two years. If it stops working, your developers can fix it themselves or replace it with something else at little cost. If the tool will be a major component of a business-critical system with a high uptime requirement, you’re better off choosing a tool backed by a stable company (or companies) and for which standard support offerings are available from a reliable provider.

Some important factors to consider when deciding whether a particular open source tool is appropriate for your purpose are:

- How critical is the function or system the component is supporting? How damaging would a bug or failure be to your business?
- Is the component relatively simple and the code easy to comprehend? Or is it sufficiently complex that debugging an issue could require an unacceptable investment of time?
- Do you have engineers in your organization who are familiar with the component, including its interfaces, functionality, and source code? Do you at least have engineers who are skilled in the programming language or framework the component uses? Are there engineers who are skilled in the programming language or framework the component uses?
- Is the open source project active? Have there been updates in recent months? Does the project have more than one or two active developers?
- Is the component security-critical? Does it expose interfaces to the internet or to your network? Are the maintainers security-conscious and do they fix vulnerabilities quickly?

Contributing to Third-Party Open Source Projects

While a company can benefit tremendously from open source software even if it never touches the code, it will leave a lot on the table if it rules out customization of open source tools or contribution of code back to the projects it relies on.

Benefits of modifying open source software

There are a number of reasons why it might be beneficial — or essential — to modify an open source component your company is using. They range from the mundane to the strategic:

- **Bug fixes.** No software is without bugs and as your developers use open source components in their development, they'll sometimes need to fix issues in the open source code to make it work for their purpose.
- **Security patches.** These are the subcategory of bug fixes you ignore to your peril (or unwelcome publicity). Whether your developers find them or learn about them from published databases, security vulnerabilities must be patched immediately, and that means modifying the source code.
- **Improvements.** Often, your developers will find an open source tool provides almost everything they need, but they need to add some functionality to make it work for their use case or environment.
- **Significant new features.** In some cases, it will make strategic sense for a project team to significantly modify an open source tool, particularly if a product or service relies upon it heavily. Google's Android is a great example: to make Linux perform adequately on a mobile phone, Google heavily modified a number of major subsystems. It took months to get the modifications accepted into the Linux project's repositories, but (at least according to Google's calculation) that was better than building from scratch or buying.

Benefits of contributing to third-party open source projects

Financial services is a highly competitive industry and secrecy is often second nature. But in most cases, your modifications to third-party open source software will return more value if you contribute them back to the project. This may seem counterintuitive at first but, unless the modifications are highly strategic or unique to your business, contributing them comes with several benefits and few downsides.

Keep in Step with Security Updates and New Features

Software products are living things. Except for the simplest applications, software will always grow and change as security vulnerabilities are discovered and fixed, bugs are squashed, functionality is optimized, and new features are added.

For the security updates alone, it's important for consumers of open source code to keep in step with changes in the upstream project. If you contribute your changes back, this is easy: your modifications will be part of every new version released. If you don't, your developers will have to incorporate the modifications into every new version after it's released by the upstream project.

Besides being a waste of developer time and attention, this means more time for malicious parties to discover and exploit your out-of-date code. Getting the changes you need into the upstream project quickly keeps your software more secure and your developers more productive.

Benefit More from Community Contributions

Anytime you enhance an open source product to better suit your needs, chances are those enhancements will be useful to another member of the community. Contributing them back to the project gives others the opportunity to use and improve those features—changes your company will benefit from.

Support the Health of Projects You Rely On

Contributions are the food that nourish open source projects. A steady stream of contributions from a robust community of users keeps a project healthy—it makes the maintainers' efforts rewarding, creates a sense of community that encourages others to join in, and it ensures that there are always candidates for project leadership in the event one of the existing leaders needs to step down.

By empowering your developers to contribute to open source projects and participate in the surrounding communities, you help to ensure that the projects you depend upon remain actively maintained and well supported.

Empower, Retain, and Attract Developers

Apart from these benefits to your software's security, functionality, and longevity, empowering your developers to participate in open source communities will improve their quality of life. Developers want to contribute to the open source projects they work with; increasingly, the most sought-after developers demand it.

Just as your developers are more productive when they can rely on readily available open source components, they preserve their time, energy, and enthusiasm by contributing back their modifications to the projects that produce them. The alternative—revisiting their prior work every time the upstream project releases a new version (or worse, failing to update the modified component)—is a recipe for frustration.

By contrast participating in open source projects improves developers' jobs in tangible and intangible ways. Beyond making the developer's job easier, giving back to the projects they benefit from feels good. This is what we mean when we talk about open source "community"—the satisfaction, appreciation, recognition, and relationships that arise from solving shared problems with peers.

Publishing Internal Software Projects as Open Source

Given the real and perceived benefits of participating in open source software projects, many companies also ask themselves if they should publish code they have written internally as open source. The motivations for doing so are numerous and covered in great detail in a recent report from Mozilla, creators of the Firefox browser and one of the leading organizations in the open source software space. While the report itself is worth reading in full—and contributing to if you are so inclined—a synopsis of the main motivations for publishing internally developed software is in order here.

Creating a de facto marketplace standard

Open sourcing internally created tools often begins with an impetus to see one's company's process for solving a particular technical problem become a de facto standard. Whereas industry once looked to various standards organizations to do the valuable work of specifying the operation parameters for various protocols, etc., publishing code as open source fulfills the same function but in the reverse way that standards creation usually operates. In the typical standards process, a technical specification is requested, created, commented upon, goes through extensive rounds of revisions, and only then is the code to prove out that implementation created. Sometimes, sadly, at the code production phase it is discovered that a particular standard fails to meet technical and market requirements, meaning the entire standards production process begins anew.

In contrast, the open source code first approach begins with a publisher producing a working implementation, thereby inviting wide adoption. Cost-free source code attracts the interest of both companies, e.g. due to development and/or licensing cost savings, and developers, who are able to seamlessly 'try before buying.' Assuming the implementation meets market and technical requirements, it can be safely assumed that broader adoption will follow, thereby creating a de facto standard as numerous companies begin using the software and contributing any features they require to ensure the implementation meets their needs for inclusion in their default software stack.

One recent example of this open source first approach to standardization can be seen in the container orchestration space. Cloud native computing has de facto standardized around the Kubernetes project, even with competitors to this technical implementation, such as Docker Inc. and Mesosphere, providing product integration with Kubernetes.

Further, creating this de facto standardizing process enables companies to create an ecosystem that encourages consumption of their fee-based products. In our previous example, Kubernetes, Google has gained a significant market advantage in its cloud computing business due to the popularity of Kubernetes. With widespread adoption of Kubernetes, Google Cloud Platform becomes a more attractive choice for organizations choosing a cloud provider, as the creator of and number one contributor to the Kubernetes project has a significant perceived quality advantage over other cloud providers. Further, with Google's move to provide support for on-premises deployments of their cloud technologies, this perceived quality advantage supports their attempts to gain customers in this space, as the large enterprises they are targeting with this new on-prem product offering are largely standardizing on Kubernetes for containerized workflows in their own data centers.

Software commoditization

In cases where a company is competing with another provider in the same technical space,

The referenced report, Open Source Archetypes: A Framework for Purposeful Open Source, can be accessed at https://blog.mozilla.org/wp-content/uploads/2018/05/MZOTS_OS_Archetypes_report_ext_scr.pdf. Mozilla and its research partner, Open Tech Strategies, encourage contributions to augment their findings by contributing to their GitHub repository at https://github.com/OpenTechStrategies/open-source-archetypes or via email at archetypes@opentechstrategies.com.

open sourcing a particular offering can provide the publisher with an economic advantage by commoditizing one portion of their ecosystem offering. By offering a free alternative to the competitor's product, the publisher can exert economic pressure on their competitor by luring away customers to their "good enough"—or even superior—implementation at no cost, thereby undercutting competitor revenue.

Further, assuming effective strategic planning and execution, the publication of this software can not only undercut competitor revenue but can also motivate customers to transition entirely to the publisher's technical ecosystem due to the stickiness of their now open sourced solution.

Enhancing marketplace reputation

Another common reason to open source software is to enhance the publisher's industry reputation. Open sourcing code signals to the marketplace that the company publishing that software is trustworthy, as there are literally no secrets about how the code operates. By demonstrating that the company has 'nothing to hide,' or even simply associating the company with the popularity that open source now enjoys across the technical spectrum, companies have a powerful tool to shape the perception of their organization in the minds of customers and would-be employees, particularly developers.

Balancing the value of IP with business value of open sourcing

One concern often raised when considering open sourcing code is the question of the value of the intellectual property that the code base represents. The company considering publication of the software has invested considerable resources in creating the solution, and releasing it under an open source license can appear to be giving up tremendous value. However, this concern can be obviated by understanding the strategic value of the code base in question.

If a particular code base is of considerable strategic value to the organization in delivering value to the customer, there must be a corresponding business value to be gained in open sourcing it. As enumerated above, this value may be undercutting a competitor's business or attracting new customers to the publisher's platforms. The key to open sourcing particular offerings without diminishing the value of the company's intellectual property portfolio is to understand whether or not the code base has greater value to the organization as proprietary intellectual property or as intellectual property now made available for wider use. For applications or technical workflows that are a key differentiator in delivering value to the customer versus the competition, open sourcing doesn't make sense; instead, there is greater value in being the only one to offer the particular solution that customers are willing to buy.

As more and more of the software that businesses rely on has transitioned to open source, businesses are finding that their key differentiators lie not in the software that they create, but on execution and differentiation in customer service.

Key differentiators lie not in software but on execution and differentiation in customer service.

For example, Allianz SE announced at the start of 2018 that they would open source much of their Allianz Business Platform, making it available to other insurance providers, while they sought to create revenue through differentiation of the insurance offerings to customers. If any of their competitors can use the very platform that Allianz uses, Allianz is compelled to deliver greater value via their consumer offerings versus any perceived value of—or lock-in to—the platform offering itself. Further, by open sourcing their platform, Allianz has the opportunity to attract more talent to the curation of the platform, including engineers who are not their own employees, leaving Allianz the ability to deploy resources that may have been invested in building out their business platform to other efforts that provide value directly to their customers.

Conclusion

You can find more resources on the FINOS website to help you shape your open source strategy, policy, and processes.

- Our Reference Free and Open Source Software Policy for Financial Services Firms is a free, open-source template for a robust and progressive open source policy and implementing processes.
- If you're interested in engaging with your peers about developing an open source program within your firm, join our Open Source Readiness Working Group, which facilitates knowledge-sharing among member firms seeking to advance their open source journeys.
- Find more at finos.org/osr.

Open source provides value across the full spectrum from consumption to minor contributions to publishing major proprietary projects. It's essential to consider numerous factors around open source software including goals, policies, processes, and support models.

With continued cost pressures and competition for high-value resources open source software is a necessity for every business. Firms can and should leverage the vast amounts of open source that make common workflows and processes more efficient and cost effective; improve developer experience; and reduce time-to-market. They should use open source to develop standards, promoting greater efficiency and shared utility across the industry, while simultaneously enabling their developers to focus on value-add offerings for clients. They should make major contributions in areas that have wide interest and potential reuse, benefitting from additional scrutiny, ideas, and resource. To sum up, there is tremendous business value in open source and financial services firms are decisively embracing this.

In future papers, we'll look at related topics in more detail including how to define an open source strategy and the place foundations hold in open source.

FINOS

Fintech
Open Source
Foundation

330 Primrose Rd. Ste. 400
Burlingame, CA 94010

www.finos.org