6 motivations for consuming or publishing open source software

By Ben Balter

December 9, 2015 | 0 Comments | 15 min read

No readers like this yet.

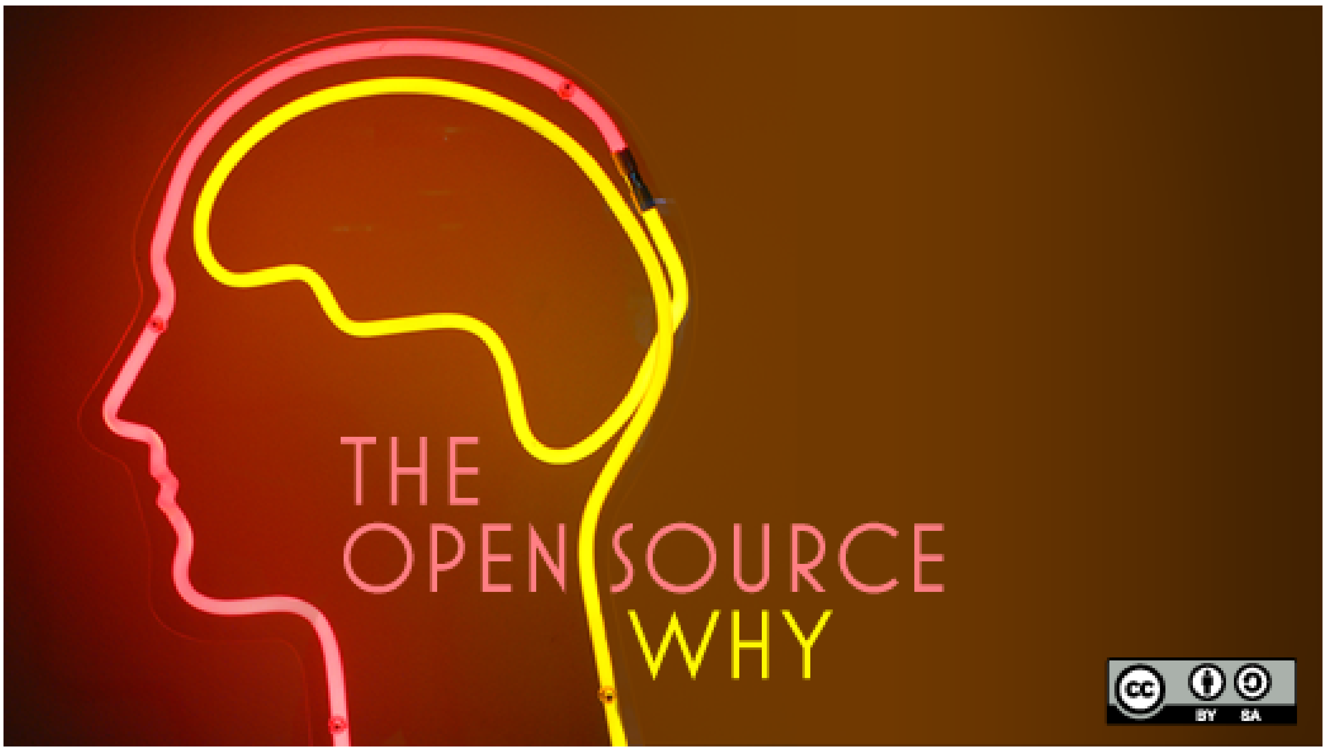

Image by: *Opensource.com*

Open source is awesome, and there are many reasons why you might consider consuming, publishing, collaborating on, or supporting open source.[1] Here are a few of them:

1. Microeconomic motivations

Open source is in your best interest, whether you're an individual, a corporation, a small business, a non-profit, or a government agency.

Shift developers from low-value work to high-value work. We like to say in open source that *all the easy problems have already been solved*. Blogging, content management, and operating systems are all problems with established (and mainstream) open source solutions, to name a few. While your developers could spend their time reinventing wheels that the open source community have already perfected, it's far preferable to use the worlds best wheel, especially when that wheel comes at no cost to you. This frees developers up to work on yet-unsolved challenges, the types of challenges that are unique to and add value to your organization's mission. Why not stand on the shoulders of technology giants?

Lower total cost of ownership (TCO): Using open source software yields a lower total cost of ownership when compared to closed source and proprietary alternatives. Adopting open source software generally has a lower up-front cost (because the software often comes at no cost or relatively low cost), and shifts the cost center from licensing (an operating expense) to customization and implementation (a capital expense). Additional costs like training, maintenance, and support are sunk costs. You're going to be paying for both regardless of if the software is open or closed source, the cost often being baked into the license in the case of commercial, off-the-shelf software (COTS). What makes open source unique is that you're not paying for the right to use the underlying intellectual property.

Cost	Open source	Proprietary
Licensing	No	Yes
Implementation	Yes	Yes
Maintenance	Yes	Yes
Support	Yes	Yes

Given enough eyeballs, all bugs are shallow (Linus's Law): Empirically, open source tends to produce better quality software than its proprietary or alternative counterparts. When you write closed source software, the only developers that can potentially detect, diagnose, triage, and resolve software bugs are those that happen to be employed by the company that publishes the software (or more likely, the arms-length contractor they pay to build it). Open source provides three advantages: first, you have the opportunity to tap the knowledge of the world's best developers, not just those on one organization's payroll. Second, the number of potentially contributing developers and thus the potential knowledge pool is orders of magnitude larger. Finally, open source software gets adapted to a variety of use cases, not just the one the publisher originally intended, surfacing bugs and edge cases much more rapidly than traditional, predictive QA processes.

Modern software development practices: Open source software is more than simply "published" code. You'd be hard-pressed to find an open source project that follows outdated, waterfall, or rigid command-and-control development philosophies, still common in many large, bureaucratic organizations. By virtue of being distributed and unbridled by policy or technical debt, open source projects all but necessitate modern software development workflows. These workflows are electronic (meaning process is naturally captured and exposed), asynchronous (meaning decisions are time and location agnostic), and lock-free (meaning contributors can rapidly experiment without prior approval). These three workflow characteristics means more rapid development cycles and more frequent releases without sacrificing quality.

Open source is the future: You'd be hard pressed to find a startup today worth it's venture capital funding not based, at least in part (if not largely) on open source. Open source isn't a fad, or a bunch of hippies out in California passing around tie-dye laptops like they would illicit substances. Open source is how modern organizations, and increasingly more traditional organizations build software. It's becoming exceedingly challenging to make the argument that five-or-ten years from now the technology landscape is going to be less collaborative and more closed. Heck, even Microsoft, traditionally one of the most stark opponents to open source, has open sourced their primary development framework (along with Apple and Swift). Today, all of the largest names in technology, from IBM, to SAP, to Adobe actively participate in the open source community. It's slightly inaccurate to say that "open source is the future." Open source has already won.

Patch on your own schedule: So long as software is written by humans, bugs and security vulnerabilities are inevitable. When a vulnerability is discovered in a proprietary software project, you have to wait for the workday to begin in the company's timezone, for meetings to be held, tasks deligated, and code written, all before G&A teams like legal, sales, and marketing have to coordinate how to get the fix into the next regularly-scheduled release. For proprietary software, bugs and vulnerabilities affect the bottom line, and thus there's a disincentive to make their details widely publicized. With open source, not only can leaner, more agile, non-profit-oriented organization move faster, since you have access to the source code, you can often apply fixes, both large and small, at your own convenience, not at the convenience of the publishing organization's release cycle.

Upstream improvements: If you consume open source software, it's in your best interest to contribute back. Contributions can be in the form of reporting bugs, or even submitting proposed fixes. Since software is written by humans, it's highly unlikely to be perfect, and even if so, likely doesn't satisfy every use case. Rather than forking the project and implementing changes into your own version (closed source), submitting bug reports and improvements upstream (open source) allows you to more easily continue to benefit from the subsequent fixes and improvements submitted by others.

GitHub Cofounder Tom Preston-Werner lays out some additional arguments in his post *Open Source (Almost) Everything*:

Force multiplier: Open source is a force multiplier for your developers. This happens in three ways: First, when communities form around shared challenges, the diversity of ideas that naturally emerges surfaces better solutions than if the marketplace of ideas was limited to just your organization, meaning developers are working smarter. Second, exposing the problem space to other interested organizations provides additional human capital to tackle your challenge, meaning the solution has more developer hours thrown at it, at no additional cost to you. Finally, "more users means more use cases being explored, which means more robust code."

Modular: Open source projects tend to be more modularly architected, improving both the flexibility, and the robustness of the code. When you're building software for a single use case, you're able to take some technical short cuts. The problem comes when you'd like to use that software in a different use case, or when your requirements change. Open source, by its nature, is built for a variety of use cases, environments, and users. This means more options (rather than hard coding defaults for a particular use), and a tendency to encourage more modularity (rather than assuing a one-size-fits-all featureset), resulting in greater flexibility and lower customization costs in the long run. Put another way, open source necessitates cleaner, more maintainable code. "Even internal code should pretend to be open source code".

Reduce duplication of effort: You should focus on your core competency. What makes you unique or gives you an advantage? Everything else is the work everyone else is also doing: or put another way, the work you need to do so that you can do the work that you want to do. It doesn't have to be that way. Open source reduces duplication of efforts, both within an organization and across organizations, by allowing for individual components to be shared. Coca-Cola's secret sauce is the formula for its syrup, not its ability publish blog posts or post press releases. Using an open source CMS, or sharing their built-in-house blog components with the world, doesn't make Pepsi taste any better. "Less duplication means more work towards things that matter."

Great advertising: Maintainers of successful open source projects are often seen as industry leaders, providing themselves with the ability to shape the conversation around a particular software problem and associating their brand with the preferred solution. 37Signals is known for creating Ruby on Rails. GitHub is known for creating Hubot. ("Within two days it had 500 watchers on GitHub and 409 upvotes on Hacker News. This translates into goodwill for GitHub and more superfans than ever before").

Attract talent: Developers want to work on yet-unsolved problems. Open source allows you to showcase to the developer community, the interesting challenges you face, and how you think about solving them. Open source developers can casually contribute to projects, to learn how you work, and what it's like developing software for a particular set of challenges. If they like what they see, there's a much better chance that they'll apply for a job, than if your organization was a black box when it comes to what it's like to work there. "Smart developers like to hang out with smart code."

> **Best technical interview possible**: Technical interviews traditionally involve working on a simulated problem that can be tackled in a set amount of time with little additional context. Such simulations, by definition, aren't real world use cases, nor do they show what working with an applicant would be like. Open source provides visibility into both how a candidate solves problems, and how they work with others. You can hire much more confidently if, for the past six months, the candidate has been contributing to the project you want them work on, and you like their work. "[T]he best technical interview possible is the one you don't have to do because the candidate is already kicking ass on one of your open source projects."

2. Macroeconomic motivations

By combining and augmenting parallel or related efforts, open source makes a society more efficient at producing higher quality software.

> **Efficiency**: Many of the microeconomic arguments above, in the aggregate, have a macroeconomic impact. When firms work more efficiently, the economy produces more (and better) software, software that can in turn, improve lives. If you believe that all the easy problems have already been solved, then on a macroeconomic level, open source allows you to move firms from lower-level work to higher level, yet-unsolved challenges. The results of this higher level work tends to be disruptive, rather than iterative, creating the churn that invigorates the economy. If every technology company must devote a few years of R&D solving the same 5-10 problems, that delays (or at the very least shifts capital from) the company's ability to be productive and produce technology that's valuable to society, not to mention, it raises the barrier for new market entrants to offer alternatives to established firms.

> **Stand on the shoulders of giants**: Today, innovative technology, the type of technology that improves quality of life, isn't created in a vacuum. Even closed source technology, from the cell phone in your pocket to the car you drive to the Fortune 500 company that produced them, rely heavily on open source (Don't believe me? Take a look at your phone's "settings -> about" page). Just as algebra and trigonometry being unencumbered by private sector copyright gave way to everything from calculus to quantum physics, open source allows developers to take already-solved problems as a given, to rely on the knowledge of experts beyond their own domain of expertise, and unlocks their potential to create new inventions, otherwise not possible.

Fuel the marketplace of ideas: Software is nothing more than technical knowledge. There was once a time when alchemists would withhold the results of their work, claiming their discoveries as proprietary. Each had to learn the hard way the result on the human body of drinking lead. When those alchemists began sharing their work, we began calling them scientists, and the scientific revolution was born. The same is true of mathematics, literature, and computer software. Two developers working individually may come up with two solutions to a given problem, but through dialog and collaboration, each bringing their own knowledge and experience, may discover three, five, or ten solutions to the same, and society is better as a result.

3. Moral motivations

The formal name for open source is free/libre open source software. As such, open source motivations have a strong moral component.

Free as in speech, not as in beer: Open source software is not without cost. When open source software is called "free," it is a reference to the rights the software consumers receive, not the cost they must pay. Adobe's Flash player, for example, is free software in the economic sense, but is still at the core of proprietary (non-free) software (and formats). Specifically, free software refers to four core freedoms: the freedom to run the software, the freedom to study and modify the software, the freedom to redistribute the software, and the freedom to distribute your modifications.
As the Free Software Movement argues, non-free software risks software that can control the user (with the publisher controlling the software). This creates the potential for the software becoming an instrument of unjust power. Today, it's not uncommon for proprietary software to spy on users (e.g., phoning home), to restrict them (e.g., DRM), to censor them (e.g., corporate firewalls), or to take advantage of them (e.g., unskippable commercials). This becomes even more important as the Internet of Things emerges, and risks turning the world into "the Internet of telemarketers" or "the Internet of snoopers." Free software places power back in the hands of users and ensures users control the software they use, not the other way around.

Obligation to give back: Open source is the give-a-penny-take-a-penny jar of software. If you consume open source, be it a server, a desktop publishing application, or a software library, you have an obligation to give back to the community. After all, without the contribution of others, the micro- and macroeconomic motivations would no longer hold true, and open source as we know it would cease to exist. This is the Golden Rule, or in traditional philosophical terms, a categorial imperitive.

Governments should give what they've developed to the people who pay for their development: If the development costs are paid by a government, then there's an additional argument for giving back. Governments take money (taxes) to perform services for their populace. For example, the US federal government was established through its Constitution by *we the people* to perform a number of important tasks. If the populace (we the people) are paying for the development of software, it stands to reason that we should receive what we paid to develop, including the software we paid to develop.

Teaching the next generation: Many of the industry's most prominent engineers today cut their teeth by learning from open source. When software's underlying code is made available for inspection, consumers can learn how their favorite software works and computer science courses can analyze how the industry's cutting edge technology is built. This goes a long way to train the next generation of software engineers (who without open source would be left guessing at the inner-workings of prior iterations).

4. Transparency motivations

Open source allows for greater transparency of process, whether that's the transparency necessary to check a government action or the quasi-governmental function of a private company's software.

Showing your work: As the government increasingly relies on technology to regulate industries and deliver citizen services, being able to see the underlying algorithms and processes are essential to checking the government's work. If a closed-source software package is used to calculate my taxes or allocate broadcast frequencies, how do I know that the process, *our process*, is fair and accurate? Whereas human processes can't be copyrighted, when close source, such processes become a black box, minimizing the potential for a citizen counterbalance.

Positions of public trust: As the software produced by private corporations are increasingly placed in positions of public trust, the transparency obligation extends into the private sector as well. Did the voting machine accurately count my vote? When given a lose-lose choice, does the self-driving car conform to community norms? For example, if a closed-source software package is used to compare DNA at a crime scene, unlike a medical expert who can be cross examined, that proprietary algorithm is shielded from scrutiny by copyright (or patent) law. Open source, at least partial open source of particular components, will become essential as private companies automate quasi-government functions.

5. Participatory motivations

Open source affords software stakeholders, both technical and non-technical, the opportunity to shape any given software development project.

Direct democracy: True direct democracy (where every citizen votes on every issue) isn't tenable given the size of most democracies. Nor have citizen been able to participate directly in most issues, due to technical constraints. Open source changes that by allowing software stakeholders to participate directly in the software development process (as software is increasingly relied on to codify regulatory and service-delivery preferences and norms). Think the process of buying health care is confusing? Open an issue. Believe the government should use more open source, submit a pull request. The White House is already doing this for many IT-centric policies like the Digital Services Playbook, HTTPS, and open data policies.

Customer feedback: Open source empowers consumers to have a combined, powerful voice in the private sector development process. Think of it like Yelp for software. Without Yelp, a restaurants is free to upset a single customer. At most, that dissatisfied patrons could dissuadee 5-10 other potential patrons. In a post-Yelp world, customers read the reviews of strangers before they choose where to go. Open source amplifies the voice of software consumers within the consumer-publisher relationship. Not to mention, the feedback the private company receives can go towards better informing product decisions and improve the overall product. Open source gives companies a direct line to their most passionate customers.

6. Personal motivations

If you're a developer (or an aspiring developer), open source can provide an easy (and free) onramp to a path for personal growth within the software development community.

> **Learn to code**: Open source is a great way to learn how to code. Want to know how your favorite website works? Clicking "view source" in your web browser can go a long way to point you in the right direction. You can read the documentation of the software that powers it, and potentially even stand up your own clone. Want to learn more? Join the local meetup for the framework or language. Better still, submit a pull request to the project to fix a small bug or add a new feature. While there are certainly time and opportunity costs involved, all of this comes without any direct cost to the aspiring developer, at least not in terms of software, and is infinitely more inclusive than doing the same within a proprietary community, especially as an outsider. There's a reason nearly every coding school today outside of traditional academia teaches open source.
>
> **It's fun**: According to Wikipedia, open source is a hobby. In fact, open source has traditionally had the reputation of being the product of hobbyists (although I'd argue that's less true today). Open source is fun. If you're a hacker it provides an endless set of ever-changing set of Rubix cubes for you to solve on weekends. Just as puzzles (both crossword and jigsaw) provide bite-sized intellectual escapes, the order and symmetry of open source can often be a rock garden of code (especially for those for which football is an embarrassing non-starter).

There are many reasons why you should prefer consuming, publishing, collaborating on, and supporting open source, and if yours isn't listed here, I'd love to hear (and add) it. Whatever your reason, it's clear open source isn't the next big thing. Open source is already here.

For a much longer, much more comprehensive, much more thoroughly researched list, I highly recommend David Wheeler's canonical paper, *Why Open Source Software*, originally published some 15 years ago (and still very much accurate).

Originally posted on Ben Balter's blog. Resposted here via Creative Commons.

Ben Balter

Named one of the top 25 most influential people in government and technology and described by the US Chief Technology Officer as one of “the baddest of the badass innovators,” and by the White House Director of Digital Strategy as “lightning in a bottle,” Ben Balter is a Government Evangelist at GitHub – the world’s largest software development network – where he leads the efforts to encourage

More about me

Comments are closed.

These comments are closed.