

Open Source Software: It Isn't Just for Developers Anymore

Zimbra, Inc.¹
www.zimbra.com

Many enterprise end-users of information technology are increasingly convinced that the existing enterprise software playing field is tipped against them. End-users seek to anoint “winning” platforms (e.g., IBM mainframes, Microsoft desktops, Oracle databases, SAP ERP, and so on) because enterprise software is so expensive to develop, debug, deploy, and integrate. Precisely by anointing winners, however, end-users have historically “locked in” those platforms, and thereby given the associated vendors more leverage (such as pricing power) than is ultimately in the end-users’ best interests.

Open standards were supposed to help the industry out of this conundrum. Interoperability via protocol standards like TCP/IP and HTTP/S has certainly been hugely successful in allowing alternative technologies to work together. At the same time, overall platform standards like POSIX, SQL, and J2EE have come up well short of making operating systems, relational databases, and application servers, respectively, into interchangeable commodities (although there is no doubt that they have protected a great deal of investment). The challenge, of course, is that software platforms have so many touch points—programming models, developer tools, administrative tools, security, and so on—that a sufficiently broad platform standard is simply too expensive to develop and enforce. Enterprise messaging and collaboration is now comparable in breadth to these system software platforms, and the collection of associated standards (SMTP, MIME, IMAP, POP, iCalendar, RSS, VoIP/SIP, IM/XMPP, ...) protect investment without yet making messaging servers into commodities.

Open source picks up where open standards leave off. Open source has a rich, multi-decade history, but is only now emerging at scale as a tool for information technology (IT) to better control costs and protect investment. Of course, open source software (OSS) is not entirely *free* even if you don't have to pay for it: deploying software inherently involves some amount of investment and risk. The promise of open source is rather about better managing that investment and risk for the users of a software product. Subject to the restrictions of open standards *and* open source, tomorrow's most successful commercial software companies might at best become 150 lb. guerrillas, instead of the 800 lb. guerrillas we face today (see below).

Historically, open source has won the hearts and minds of developers first: hugely successful open source technologies like Linux and the Apache web

¹ Zimbra is a trademark of Zimbra, Inc. All other trademarks belong to their respective owners.

server were brought into the enterprise by the programmers that used them as the foundation for new applications. More recently, we have seen the emergence of open source applications, such as Asterisk and SugarCRM. However, even these end-user applications are still often most strongly advocated by developers, because of their desire to customize and extend application functionality. Enterprise messaging and collaboration fits somewhere in between: solutions like the Zimbra Collaboration Suite (ZCS) combine systems- and application-oriented software. End-users may seek to customize or integrate Zimbra, but at the same time, ZCS is most often deployed as is “out of the box.”

However, even if you the end-user have *no intention* of ever enhancing or augmenting an end-user application like Zimbra, there are still *substantial benefits* to choosing open source technologies. Here are our top ten reasons to favor open source over proprietary alternatives:

1. **Avoid “buy and rent”** – Under the traditional enterprise software sales model, you pay (often dearly) for an end-user license agreement (EULA), and then immediately start incurring annual fees for maintenance, upgrades, and support. With OSS, you typically just pay for the latter.
2. **Backload costs** – EULAs are typically front-loaded in the sense that you pay to license the software (often across the whole of your enterprise) in advance of that software proving its value. With OSS, you can “backload” more of the costs by trialing technologies without fees, and only paying support/maintenance after success is assured.
3. **Optional maintenance fees** – Maintenance contracts for closed-source software are typically mandatory for any business-critical deployment. Moreover, you sometimes find you are paying a vendor for the privilege of debugging their software (and often in the process teaching the vendor’s support personnel about their own product). With OSS, you only pay for service that is of real value, since there are alternatives ...
4. **Competitive maintenance pricing** – Open source allows third-party vendors to competitively bid for your support contracts, ensuring that even the originators of open source intellectual property must compete effectively to win your business.
5. **Feasibility/cost of customization and integration** – When you invest in adapting software to meet your needs, you are justified in demanding ownership of the resulting IP, as well as the ability to apply that IP to future releases or in conjunction with other products. While it remains technically challenging to apply customizations to new software releases, the OSS model gives you the most viable model for doing so.

6. **Greater technology investment as a share of end-user fees** – According to Goldman Sachs, 76% of a traditional proprietary software vendor’s revenues go to sales and marketing. Instead, you might prefer your suppliers invest more of your IT expenditures in product R&D, quality assurance, and customer support.
7. **Long-term viability of deployed software** – With OSS, you always have the source code, and so early adoption does not entail the same risk as that associated with building on top of non-established closed-source products. (Of course, you still want to choose “winning” OSS projects, products, and companies in order to get the highest return on your investment.)
8. **Help from the open source technical community** – Historically, commercial software vendors have had to become quite large to be able to foster the informal technical communities that spring up around successful software products. Microsoft, for example, has done an exemplary job of facilitating technology adoption by “darkening the skies” with CDs and filling the shelves of the high-tech bookstores. However, the combination of the World-wide Web and open source permits very rapid scaling of these essential technical communities “on the cheap”. These communities are empowered to facilitate debugging, extension, customization, and localization (such as via the translation of message catalogs).
9. **Community innovation** – More importantly, successful open source communities accelerate innovation by attracting *and* empowering a broad pool of thinkers that contribute ideas as well as code. At the same time, such collective innovation must be balanced by the need for product coherency—successful open source projects, whether they are rooted in companies or independently, have strong hierarchical leadership that chooses the best-of-breed extensions for incorporation in the product (often by considering the real-world success such extensions have had in the marketplace).
10. **Increased security and reliability** – Perhaps most importantly, the open source model is a powerful asset in the pursuit of higher-quality software, which is the topic of the next section.

Open source and quality

In the pursuit of software quality—reliability, ease of use, security, and so on—open source is an *asset* rather than a liability. If this point remains controversial, it is in part because of the fear, uncertainty, and doubt raised by vendors that are competing with open source. The reality, of course, is that

most software (open source as well as proprietary) is of insufficiently high quality from the end-user's perspective. After all, the QA investment required to get from innovative software project to widely-adopted, business-critical software product (with fault tolerance, security, integration, compliance, performance, scalability, usability, ease of configuration, ease of administration, ...) is high enough that few software projects (open source or proprietary) ever get there.

But some open source software clearly has cleared this hurdle. Take Apache, Linux, and Firefox for example, each of which is of defensibly higher quality than its proprietary alternatives. Perhaps the intended criticism being made by the proprietary software vendors is simply that open source software is cheap enough (free) that the associated QA regimen must somehow be inadequate. Closing this is a gap is precisely what the commercial open source (not an oxymoron) vendors (e.g., Red Hat, Mozilla Corp., Zimbra, but also IBM, Oracle, and Apple) have as their *raison d'être* (see below)! But more importantly, this argument discounts the QA contribution of thousands of talented community members that are testing and troubleshooting the software in advance of end-user commercial deployment.

Of course, not all open source software reaches such quality "critical mass", but for the ones that do, open source is an asset in the pursuit of software quality:

- **Scrutiny of the source code** - To ensure software quality, there is nothing like the added developer accountability that results from the public scrutiny of (and commentary on) his/her code base. Products that pass muster are likely to have cleaner, more elegant internal architectures that will make them a better long-term investment. (The reverse is also true—vendors of proprietary products often fear that open source will air their architectural dirty laundry, as much as they fear it will cannibalize their license revenues.) It turns out that this is true even for security—public scrutiny of the source code has actually helped Linux, Apache, and Firefox strengthen their network security far more than it has helped attackers identify potential weaknesses.
- **Transparency** - Open source projects/vendors are far more likely to open up their bug databases, provide open communication between developers and users (via forums), as well as to open their development, build and test regimen. By inviting users into the "sausage factory", open source is at least welcoming greater participation and feedback on quality.
- **Continuous quality assurance (QA)** - Most open source software is developed in public in that new code for the project is typically posted to the Internet daily or weekly. The fact that community members (as

well as anyone else with an Internet connection) can download and use this forward-development code demands a higher-level of automated QA on the nightly and weekly builds. In this way, quality assurance is built into the development process rather than left to the end of the “new feature” coding cycle.

Intellectual property (IP) hygiene - Most software (both open source and proprietary) is now built from existing components. There is arguably now less risk of IP contamination (that is, does your vendor own the IP rights to the software they are giving you?) with open source than with closed source alternatives, because all such open source dependencies are transparent.

- **Community QA** - Last and most importantly, successful open source projects are able to draw large developer and testing communities for more economically than proprietary vendors can. (It is after all a trade—open source projects share their core software for free precisely to draw the value-add of this greater community.) The collective validation and testing investment made by the respective communities are what make Linux, Apache, and Firefox arguably the most secure and highest quality technologies in their respective categories.

In summary, software quality varies more between projects/products than it does within categories (OSS versus proprietary). So users must continue to do due diligence in order to ensure that the quality of the candidate software is sufficient for their needs—the more business-critical the deployment, the greater the need for picking winning technologies whose risk is low because quality assurance is broadly amortized. However, between comparably successful proprietary and open source alternatives, open source is likely to both provide higher quality for the price and, given the above, higher quality in general!

Why commercial OSS vendors?

While the case for open source is compelling, thus far more OSS efforts have been focused on delivering component technologies to developers than on delivering finished products to end-user systems administrators. Of the 120,000 open source projects in the world, only a subset have achieved critical mass as stand-alone, finished, and widely-adopted end-user products: these include Linux, the Apache web server & Tomcat, Firefox, MySQL, PHP, and so on. In each case, commercial ventures have sprung up around these technologies to help ensure their readiness for enterprise business-critical deployment.

Why? The hurdles for complex enterprise-ready products simply remain very high: it takes substantial investment in usability, ease of configuration, ease of

administration, performance, security, fault tolerance, configuration testing, robustness testing, scalability testing, integration testing, compliance testing, and so on. Many of these costs can be amortized across the greater OSS community, but you can continue to expect commercial entities like Red Hat, the Mozilla Corporation, MySQL, and now Zimbra to fill this gap between successful developer-friendly technology and successful end-user business-critical product.

Since most OSS that finds its way into IT environment is not itself a “core” differentiator to the end-users’ business (e.g., most end-users don’t differentiate against their competitors based on whether they use Red Hat or SuSE Linux), it makes sense to outsource the associated product engineering (via service contract) in a way that the costs are amortized across many end-users. Commercial OSS vendors are “commercial” entities—just like IT end-users, they seek to use their smarts to compete, profit, deliver ever higher value, and so on to their customers. What separates commercial OSS vendors from their proprietary peers is their commitment to the open source model, which ensures that they also deliver all of the OSS end-user benefits enumerated above.

Open source and the shrinking software industry?

The other concern occasionally raised with respect to open source (as well as other new technology waves) is that it may somehow be bad for the greater software industry. To wit: If open source, Software as a Service (SaaS), and Web 2.0 technologies are indeed driving down the total cost of ownership (TCO) of software, then won’t that ultimately cause our industry to shrink? If software unit costs (whatever the unit—users, servers, etc.) go down, then doesn’t our industry as a whole have to shrink?

- **No, because we can make it up in volume.** Microsoft became the richest computer company in history, by simultaneously lowering unit costs and increasing the overall reach of software. Open source is so exciting because the industry now has a distribution model that will ultimately deliver software beyond the current reach of Windows and Office (e.g., Nokia’s phone browser has recently open sourced). SaaS (also known as on-demand) has also well extended software’s reach--- Yahoo!, Google, and MSN are all SaaS sites after all. SaaS isn’t just about consumers either: Salesforce.com has had the most success to date not displacing Siebel/Oracle, but rather getting CRM into the hands of small companies that were still trying to do it on spreadsheets.
- **No, because we can make it up with innovation.** If all we in the software industry collective achieve in the next decade is rebuild new open source, on-demand, and Web 2.0 versions of the same tired

software from the prior decade, we deserve to shrink! The way for the industry to grow is not to set our sites on redoing what has already been done, but rather on doing new things and on doing the old things dramatically better. The best news of all is that the barriers to innovation have never been lower: new software ideas can be far more quickly realized via the re-use of open source building blocks and by *mashing up* existing Internet services.

So the answer, as always, is that the winning vendors (open source and proprietary, on-demand and on-premises, etc.) are the ones that best figure out how to deliver value to their customers. Business models and buzzwords aside, the companies that innovate to deliver higher value will thrive, and the ones that do not deserve to shrink.

Open source and intellectual property (IP) ownership

It is increasingly common practice in open source communities to insist upon a single organization owning the IP rights for a particular project. For example, the Apache and Free Software Foundations do this as well as private firms like Zimbra and MySQL. The primary goal of IP ownership is to better guarantee "squeaky clean" IP (that is, that all contributed code is unencumbered by any hidden IP rights) for the benefit of the community as well as for end-users. This also ensures that open source licensing upgrade decisions can be made for future releases of the open source license itself—without an organizational owner, projects can effectively get locked to a particular version of a license because there is no easy way to get all of the IP owners together to make a decision about upgrading.

For a potential community member, the most important thing is the long-term guarantee of their freedoms: (1) Open source rights (granted under the open source software license) to use the product in perpetuity for free, to produce and redistribute any derivative works thereof for free, and so on; and (2) rights to do whatever they may see fit in perpetuity for any of their own contributions that they made in good faith to the project. (The good contribution agreements grant contributors back all of their rights **except** those that could interfere with the community, such as the right to withdraw their contribution in the future, the right to charge the community patent royalties down the road, etc. For example, Zimbra's Contributor Agreement can be found [here](#).) This practice seeks to strike the right balance between preserving individual freedoms without sacrificing the freedoms of the community as a whole.

Both non-profit and for-profit open source organizations equally guarantee these freedoms to their communities, the difference being that the for-profit ones (like Zimbra) also strive to sell **optional** value-added services on the side,

much of the proceeds of which then funds the further development of the open source software.