O'REILLY
ONLamp.com
LAMP: THE OPEN SOURCE WEB PLATFORM
Published on **ONLamp.com** (http://www.onlamp.com/)
http://www.onlamp.com/pub/a/onlamp/2006/01/12/no_oss_community.html
See this if you're having trouble printing code examples

There Is No Open Source Community

by John Mark Walker
01/12/2006

Conventional wisdom says that powerful individuals drive open source by working against the grain to institute a methodology of sharing that would balance the power between software vendors and users.

While this makes for an entertaining narrative, there is quantitative evidence to the contrary. The reality is that placing too much emphasis on individual players in the open source movement ignores overarching economic trends that drove open source development and adoption.

Furthermore, taking the position that individuals have pushed open source forward leads to the conclusion that a core group of ideological "believers" is necessary for the continued success of open source software. Businesses unaware of the falsehood of this claim are too afraid of running afoul of the "open source community" and sometimes make decisions that are not in their financial interests.

Both open source-based and proprietary software vendors need to challenge these assumptions. An updated open source mentality has profound implications for businesses looking to leverage open source in commercial ventures. Reevaluating the open source equation in economic terms presents a different takeaway. The commoditization of software and a gradual, long-term reduction in price have played far more important roles than previously recognized. Business strategy designed to leverage open source should focus more on economies of scale (in terms of user and developer bases) and less on pleasing a mythical, monolithic community.

Some software vendors believe that open source is an ideological movement. This paradigm ignores the impact of software prices shattered by zero-cost distribution and global collaboration capabilities, both of which the internet fuels. It also ignores one of the primary factors driving customer adoption: rebellion against vendor lock-in. By combining lower cost of production with the additional freedom and flexibility endemic to open source deployments, one sees two dynamics driving both adoption and production. The push of software commoditization and the pull of customer demands have created a perfect storm for open source software.

This new perspective has implications for other areas, such as TCO and potential legal obstacles to open source. I argue that TCO is largely irrelevant when one takes the larger view of open source evolution. As for legal pitfalls, we believe that economic principles prove this fear is largely unfounded and that any legal impact on open source from patent infringement or copyright violations will be limited in scope. Regardless of who leads the charge, what legal

obstacles are thrown in its path, or whether there is any provable TCO advantage, open source will continue to expand its grip on IT.

The Conventional Wisdom

Open source conventional wisdom tells a tale of good versus evil, David versus Goliath, in a struggle to protect users from the malevolent intent of large software companies. The narrative usually begins with Richard Stallman, upset with printer manufacturers releasing binary-only drivers that prevent him from fixing bugs in the software. From there, the story includes the founding of the GNU project and the Free Software Foundation (FSF) as a means of ultimately producing a free operating system.

Conventional wisdom recognizes this as the official birth of the free software movement, an idealistic and political movement that specifically sought to protect the freedoms of computer users. This is in contrast to the open source movement, which pitched open source software production as a practical means to better software. The GNU project was working on a free operating system kernel--the last piece of the free operating system puzzle. Unfortunately, the project found itself stuck on a microkernel architecture that proved unwieldy from an engineering standpoint. Then, in 1991, Finnish computer science student Linus Torvalds wanted to run Unix on his PC. He blindsided the FSF by producing a free operating system kernel called Linux that meshed with the other GNU software to produce a working Unix-like system.

Linux and GNU continued to grow, drawing interest from an increasing number of free software contributors. Somewhere along the way, the BSD operating system became unleashed from its AT&T shackles, making for a more mature competitor eager to capture the Unix-on-PC enthusiasts. The BSD process proved to be less open to collaborators, and it frowned on free software idealism; thus more developers flocked to GNU on Linux, making it the undisputed heavyweight in the PC Unix space. In the latter half of the 1990s, some contributors grew uncomfortable with the "free software" name, due to both confusion over its meaning as well as uneasiness from its idealistic underpinnings. These contributors were in favor of emphasizing the pragmatic aspects of free software development without all the GNU baggage inherent in the term *free software*.

A major part of open source lore is the famous meeting in 1998 that included Larry Augustin, Tim O'Reilly, Eric Raymond, and others at O'Reilly headquarters. The group there coined the term *open source*. Aside from drawing the ire of Richard Stallman, who wanted to emphasize the freedom of free software, the term proved a raging success among developers looking to participate in open software collaboration. Eric Raymond wrote the essay "The Cathedral and the Bazaar," which influenced Netscape as it deliberated releasing an open source version of its web browser software.

The open source bandwagon gained steam, with a core group of supporters pushing for its continued growth. Spokespeople such as Raymond and Bruce Perens helped lead the charge with articles and appearances in various media outlets. Developers such as Torvalds and Brian Behlendorf pushed open source forward on the engineering front, with the Linux kernel and the Apache web server often cited as open source bellwether projects. These two projects have continued in this role to this day. Large high-tech companies such as IBM helped legitimize both and made them more enterprise-worthy.

Despite the nascent success of open source software, there has been increasing concern about potential pitfalls, such as patent infringement claims from large software companies including Microsoft. Many fear that Microsoft, often seen as an enemy of open source, is looking for the right opportunity to spring

a patent infringement trap. Further fueling some of these fears is the copyright infringement by the Linux kernel claimed by SCO when it filed its lawsuit against IBM. While largely seen as unfounded, SCO's claims have led to some open source leaders calling for such things as more audits of open source code and legal indemnification from open source software vendors.

Open source supporters have rallied around the cause, looking to prevent frivolous lawsuits against open source developers that might have a chilling effect on open source's momentum. Another concern is the habit of some media outlets to lump together open source with warez and piracy groups. This could conceivably besmirch the credibility of open source philosophy and those associated with open source software. Altogether, while open source and free software have made tremendous strides over the years, strident supporters see the continuing need for vigilance in order to protect what they have built thus far.

What's Wrong with That?

What if you discovered that everything you ever learned about open source growth was wrong? What if the narrative that pitches open source in terms of battling evil software giants wasn't actually correct? What if you learned that the recognized leaders of the open source movement were simply figureheads of a process already well under way? What if you learned that open source was neither good nor bad, but simply the manifestation of decades-old economic trends? What if companies mining the open source vein aren't taking the high road but rather ruthlessly applying a competitive advantage?

Forget what anyone has ever told you about open source, and learn this term: *economies of scale*. This is what has fueled the open source phenomenon. Most people recognize the role that the internet has played in terms of creating an environment capable of supporting massively parallel distributed collaboration. Too few have recognized the crucial role played by the internet in creating the economies of scale necessary for the proliferation of open source software. Furthermore, there has been little recognition in open source circles of the role the internet has played in driving down software production costs and thus software prices. It is this drastic reduction in price that is necessary for an open source-friendly environment to emerge.

The New Open Source Growth Perspective

It's the internet, stupid.

What is it about the internet that drives down production cost? There are multiple avenues to the result of reduced software cost. Foremost is the advent of the internet as a software delivery mechanism. With the internet, the possibility of practically zero-cost distribution was viable for the first time. Furthermore, digital delivery obviates physical packaging, transportation costs, and reliance on resellers that buy at wholesale, as opposed to direct customers who pay full retail price. Zero-cost distribution gave early adopters a competitive advantage over vendors that relied solely on packaged distribution. It also allowed vendors a much faster way to bring products to market.

No longer did software vendors have to allow for a certain amount of time before packages hit store shelves. When the internet became viable for software distribution, vendors could sell a product almost as soon as a release was certified for public consumption. So far, however, this establishes the internet only as a driver of reduced software costs. That alone does not link the internet to open source proliferation.

This brings up the second ramification of the global internet: instant collaboration across national boundaries. When Linus released version 0.01 of the Linux kernel to the masses from FTP servers in Helsinki, it wasn't long until developers downloaded copies in cities outside of Helsinki and in lands far away from Finland. Users of software were able to take a copy, use it, and then report back to Linus

much more quickly than ever before. Gone were the early days of the GNU project, when those that wished to use the software would order tapes and wait for them to arrive in the mail. Reporting bugs and usage issues and distributing patches back to the GNU project also was no small matter. Thus, the internet sped up development time by facilitating almost instant feedback from users regardless of location, as long as they had reliable internet access.

The distributed aspect of the internet also allowed for a much larger audience than before. If Linus had released his kernel just five years earlier, practically no one outside of Helsinki, much less Finland, would have even known of its existence. This is not to say that Linux would not have survived, but its growth would certainly have been slower. It is this massive number of users that makes the numbers work out for large open source projects.

The numbers work out because enough users report back to the Linux kernel developers on bugs or usability issues. A percentage of users contribute patches. Other users take a chance with bleeding-edge kernels not yet ready for release. It is a given that with any software project, most users will be just that—users of the software. However, a certain percentage will become beta testers, bug reporters, documentation writers, or full-blown members of the development team. This number of contributors will grow only if the overall user base grows. Without the internet, this growth of users simply doesn't happen.

Imagine the amount of money that people would need to spend if the internet were not available in order to win over the same number of users and contributors. First of all, using the Linux kernel as an example, Linus would need to print thousands of copies of floppies (remember, this was the early 1990s) and distribute them. The obvious choice would be to distribute them locally on campus in Helsinki. Then, to replicate the experience of the internet, he would need to devise some way to get them into the hands of users around the world—not just any users, but interested users. Using this example, it becomes clear why large-scale open source development did not happen until the internet came along. It also becomes clear why software engineering was largely the domain of companies that could afford the marketing and outreach required to build any type of community around technology.

So far, I have demonstrated only how the internet added cost-effectiveness to software production, but that's still not the whole story. To take the argument one step further, I will demonstrate why economic trends kicked off by the internet have made open source software not only possible, but also actually necessary for survival for some developers and software vendors.

The Rising Tide and Economies of Scale

As a thought experiment, think back 20 years to a time when the internet was still a DARPA project and the web was but a glimmer in Tim Berners-Lee's eyes. At that time, someone who could create software or build computers was pretty special. In fact, unless you worked for a software vendor or attended class in a computer science department, programming was pretty much a black art understood by an elite few. There were some computer users, but computer hobbyists weren't exactly mainstream.

What has changed since then?

If you guessed that there are lots more people now who understand at least a thing or two about computers, you would be right. Today, more people know how to use computers, program computers, and network a group of computers than ever before. What has made this possible? The internet, of course.

It's actually quite simple. With the advent of the internet came the increased usefulness of owning a computer. With more computer owners came a deeper understanding by a broader group of people of how the things worked. Of those who understood computers, a certain number became savvy users. A

smaller number became computer programmers. With an ever expanding internet, all of these numbers have increased over time. From the United States to Europe to developing countries in Africa and South America, more people understand computing than ever. The collective knowledge of computer users around the world dwarfs by orders of magnitude the entire such knowledge base of 20 years ago. That has serious ramifications in terms of how software is produced.

For one thing, software knowledge comes more cheaply than ever. As with any currency or commodity, when there is a seemingly endless supply, the price goes down.

I will focus on two commodities: software knowledge in the form of a programmer and the amount of software available for use. For the first, unless a programmer has a specialized bit of knowledge in an area of high demand, chances are that the ability to command a high salary is less than it was a few years ago. There is a plentiful supply of people who have at least some programming knowledge. Granted, especially talented programmers are always needed and can demand extraordinary compensation. However, focusing on the general case, the price of a programmer commodity has declined over the years, thanks in large part to the internet.

The spread of the internet has had a direct impact on the spread of programming knowledge. The bar to entry for a budding programmer is lower than ever, thanks to the readily available supply of tutorials and other documentation geared toward those who wish to design software. This shared collaborative medium allows programmers to instantly post new material, thus raising the collective programming IQ of internet denizens. Consequently, not only is the breadth of developers more extensive than ever, but the amount of knowledge per developer has also increased. At the moment, the internet has made programming knowledge a boundless resource with unlimited growth potential.

As for the software itself, there is more of it than ever before, much of it general-use desktop or server software for a PC. Software developers are no longer masters of a black art, and software is no longer black magic. It is an abundant commodity with an expanding breadth and complexity. A new feature for a software project posted on the internet increases the overall complexity of the project. Just as programming knowledge expands both vertically and horizontally, pieces of software wind their way to more places than ever, with the complexity of these pieces also increasing over time. A rising tide floats all boats. In this case, it's the tide of the collective knowledge base, fueled by the internet, and the boats are software developers and software projects. It is now easier than ever for a programming neophyte to search the internet, find a programming tutorial to his liking, and start building software.

This continuously evolving collective knowledge base carries another consequence: speed of innovation. Because of the speed with which users, developers, and companies can post documentation, patches, or new software projects, the product life cycle has shortened considerably. Software vendors must work harder than ever to stay ahead of the floating software boats. This constant drive for innovation means that products released just yesterday lose value more quickly than before, due to future products already filling the software pipeline.

What are the implications for software developers? The obvious manifestation of a lower bar to entry coupled with an increasing number of programmers is that it is getting awfully hard for a developer to charge for software. (Quick, tell me the last time you paid for a bare-bones email client.) It used to be that a developer could hack up some small utility, pass it around as shareware, and ask nicely for people to send money. While shareware still exists, the trends are not in its favor. More recently, people who hack together a simple utility simply give it away. They don't ask for payment, because they recognize that it's generally a fruitless endeavor. It's not that they give away the software because they think it's a nice thing to do; they give it away because it's the only way anyone will actually notice.

The general case of a broad, horizontal software market has now arrived to the point where a healthy open source ecosystem is possible. The price of software is asymptotic to zero, there is an abundance of users and developers, and there is a vast collective knowledge base that empowers users to harness the power of available software and produce their own. In this situation, it is impossible to create

general-use software for which you may charge a premium price.

With prices approaching zero, software developers have two choices when trying to win over users: (1) add features not available elsewhere, and (2) release the source code. There is no other currency of value that developers can extend to users. In fact, these two options are actually interrelated. A software developer trying to accomplish option 1 on his own will face a daunting task, whereas a developer who releases source code, assuming the project is viable, will have a ready supply of suggestions for improving the software and adding features. She will also have a cheap and fast means of distributing the software complexity around the world, with the implied deepening of the collective software knowledge base. This collective software knowledge base will pay dividends in the form of knowledgeable users and contributors.

When faced with these two options, a developer who chooses route 1 with a traditional software development approach will actually find himself eclipsed by a developer who chooses option 2, even if the ambitions of the latter project are lower. Granted, only a small percentage of the user base will ever contribute anything, but this thriving ecosystem of users and contributors is what will make a software project viable over time. Thus, in a system where software prices approach zero, open source becomes a necessity in a competitive market. (I plan to discuss the debate on leveraging this model for profit in a follow-up article.)

Without prices that approach zero, there is simply no room for viable open source options. To test this, think of most vertical software markets. These are the homes of specialized software tools of which only a few users are intimately familiar. Without a broad base of users and a deep collective knowledge base, there is considerably less downward price pressure. Consequently, there is simply no incentive to release open source software in those markets.

When taking stock of vertical software markets, I notice a decided lack of open source alternatives to commercial software. One could test assertions made in previous paragraphs by looking at vertical markets that have recently broadened in scope following an increase in the sheer number of inhabitants in that market. If the above assertions are true, there should be an emerging open source ecosystem in that market, albeit less mature and feature-complete than competing commercial products.

Conclusion

I have shown how the internet has driven software prices into the dirt, created an environment conducive to open source collaboration, and provided the infrastructure for that collaboration to actually take place. I have also shown how cheap commodity software markets are necessary for open source development and how open source is not viable in less mature software markets without the necessary economy of scale. When viewing open source development from this perspective, some things become clear that perhaps were not before.

1. The continuing expansion of the internet is necessary for continued open source proliferation.

   In order for more projects to grow in a vibrant open source ecosystem, there needs to be a fresh supply of new users and developers. The economies of scale that spawned open source development need to keep expanding, or else there is a risk of stagnation.

2. Given current trends, open source will continue to expand in scope, prevailing in more markets.

   All signs point to an expanding internet for the foreseeable future. This means that the trends that result in cheap software commodities should maintain their steady pace. As such, the open source footprint should continue to expand.

3. There is no open source community.

   Looking at open source from an economic perspective, it becomes clear that Linux or its equivalent was bound to happen eventually, regardless of whether Linus decided to release a kernel in 1991. The same applies for Apache and any other project. Both of these are the natural result of massive price drops in their respective markets. The view that there is a core group of altruistic companies and true believers driving open source forward is simply false. The view that open source participants are idealistic Davids fighting against software Goliaths is also false. In fact, surveys of open source participants tend to bear this out.

4. Open source is neither good nor bad.

   Open source is not a religion. It is not an ideology. It can be used for both good and bad. It does not inhabit the higher moral ground, nor is it a more ethical way to conduct business. It just is, and it will continue to grow and expand.

Coming Up

In the next article in this series, I will show the other side of the equation: the customer pull on open source. I'll also demonstrate how and when a software vendor can leverage open source economies of scale in its favor, why TCO is irrelevant, and why there is little to fear from legal battles.

John Mark Walker has burrowed deep inside the bat cave, working feverishly to make sure that LinuxWorld learns how to party like it's 1999.

Return to ONLamp.com.