Open Source and the Commodity Urge: Disruptive Models for a Disruptive Development Process

MATTHEW N. ASAY

Director, Linux Business Office, Novell
Entrepreneur in Residence, Thomas Weisel Venture Partners
Founder, Open Source Business Conference

Abstract: *Open source hastens software's natural trend toward standardization/commodification. While technologically innovative companies will always find ample customer interest, the most important innovations for the next decade of software will come from business model innovation, mostly spawned by open source license requirements. Open source builds a new intellectual property regime centered on the source of code, not source code. Protection, in other words, shifts to "owning" the code creator, rather than the product she creates. Those business models that acknowledge this and leverage it will yield better profits than those that attempt a half-way embrace (or rejection) of open source.*

Introduction

We are missing the point. Yes, open source imposes dramatic changes on the software industry and, yes, it is roiling the fortunes of many an established vendor. It will continue to do so, and at an increased pace. Yet despite the sometimes anguished, sometimes giddy reception that open source has provoked in the IT world, open source is not novel. It is not odd.

Open source is simply the software world's mechanism for becoming just like everyone else.

All the world's a commodity, or services to support and distribute commodities. This book that you are reading. The chair that supports you. The restaurant you will eat at tonight. Everything, including, increasingly, software, thanks to open source. Open source accelerates the natural progress of software toward commodification, or standardization.

It is critical that IT vendors understand this so that they can deploy (or fight, if they so choose) open source effectively, and more intelligently choose how and where to innovate. Open source does not destroy all value in software innovation; instead, it shifts the control point from the code itself to the creator of the code. In so doing, open source software will not pillage all closed source software. As in other industries, there will continue to be plenty of room for upmarket vendors (e.g., Whole Foods in grocery retailing; Starbucks in coffee; and Nordstom in retail clothing).

That said, there is no room for middling and muddling. Open source will commodify from the bottom-up while "upmarket" vendors will dominate "up-the-stack." Everything else will be

a wasteland. Just as Safeway[1] finds itself pummeled by Wal-Mart and Whole Foods so, too, will middle ground IT vendors find themselves grasping at a dwindling market opportunity.

Open source offers hope, but perhaps not for the reasons normally associated with it. Much has been made about the open source revolution, and with good reason. But perhaps the best reason has little to do with development of source code, and instead has much to do with distribution, marketing, and sales. In other words, what we thought was a software development methodology may have far more importance as a business strategy that undercuts competitors while driving down costs and shifting control to buyers. In such a world, those who understand and leverage open source commodification (or escape it) will thrive – everyone else will be marginalized into economic oblivion. Commodification, the highest stage of capitalism; open source, the highest stage of software.

A Brief History of Software

Once upon a time, software did not matter – hardware did. Software was something that hardware vendors wrote to help them sell hardware. Little more. Software was important because it made hardware operate, but customers understood that they were paying for hardware, and not the software that ran on top of it. (This is still somewhat true of certain areas of the embedded software market.)

As hardware commodified, software grew in importance as a differentiator with these same customers. Not all hardware commodified at the same pace: Solarais servers, for example, handled a workload in a way that commodified hardware could not, and Sun consequently charged a premium. But the real premium increasingly gravitated up the IT stack to the applications that people ran on their hardware. Hardware was important, but only because the applications had to run on something. With the rise of Dell and other commodifiers, however, IT buyers came to care less and less about the "guts" of their computing experience – they bought systems for the applications they could run, for the productivity they could achieve.

No single company did more to send the applications trend into hyperdrive than Microsoft. Microsoft out-Apple'd Apple's Macintosh by making software easily consumable by and affordable to the masses. By simplifying computing, and by doing so at a dramatically lower cost, Microsoft grew the market by competing against non-consumption and under-consumption, inviting multitudes of average users into the hitherto closed world of computing. Microsoft's Visual Basic lowered the bar of expertise to be a proficient developer, and its Office suite created a market of home and business users who suddenly could create brochures, quality letterhead, etc. Whatever open source developers' feelings about Microsoft, they should acknowledge Microsoft as the natural parent to their own brand of commodification.

For this is what open source is doing: commodifying software. We are now at the point where mainstream software is becoming commodified by the open source community, perhaps

---

[1] Safeway is a large, US-based grocery retailer. Whole Foods, also based in the US, is an upscale grocery store that focuses on premium, organic food. Wal-Mart is, well, the world's most dominant retailer. By the time you read this, they will probably be selling cars in addition to groceries, clothing, electronics, online music, and various other product categories.

pushing all value to the services that support hardware and software. Microsoft, in a sense, is being out-Microsofted.

In this world, customers benefit as vendors focus on solving their business problems, rather than innovating new methods to achieve customer lock-in. Much of today's IT world is composed of expensive, monolithic software "solutions" that end up creating complexity and integration problems, rather than resolving customer problems. That is, today's IT industry is a morass of conflicting standards, complex installations, tepid product interoperability, and crushing expense: each a product of the industry's "whatever sells" mentality in its adolescent years. Increasingly, however, customers are tired of subsidizing the disarray, and are turning to open source as a way to get more for less. As open source proliferates, the cost of infrastructure software[2] will plummet, freeing up resources that the CIO can spend on resolving application requirements up the stack.

Vendors, for their parts, also benefit from increased use of open source, because it removes the "IP safety blanket." This is not to say that most vendors like what is happening to them. They do not. It is disruptive and, hence, highly uncomfortable, to the vast majority of traditional software companies. In the long run, however, they benefit, and will be grateful for the short-term disruption.

Such is the result of open code. Because code is open, vendors must find innovative ways to satisfy and "lock in" customers. Copyright and patent are fine for this purpose, but they pit the vendor in an adversarial relationship with the customer, whereas open source control mechanisms tend to force vendors to win by intimately understanding and fixing their customers' business problems. In addition, this commodification of IT will push vendors to move up the stack (and off the stack, into services) to deliver increased customer loyalty/value. Finally, as prices for software drop to match the drop in hardware costs, more buyers will enter the market, increasing the size of the market. Everyone wins, but more detail is needed to show why this is so.

A New Brand of Intellectual Property Protection

To fully appreciate this trend, it is critical that we better understand the intellectual property regime powering the open source revolution. Intellectual property (IP) law has always been about control. That control benefits creators by holding off would-be competitors long enough to allow the creator to attempt to profit from her innovation. I write a piece of software; I copyright it; I sell it (assuming it is a useful piece of software and I have adequately marketed it so that people know about it). Simple. This has been the software industry's dominant model for decades, and has created a few mammoth software companies that have successfully exploited their IP to generate billions of dollars in revenues. In this model, exclusion (i.e., the ability to keep competitors or customers from copying one's code and

[2] I define "infrastructure software" very loosely as the system software below the application layer of the software stack. It is, in other words, the "infrastructure" on which the applications run. This would include such software as operating systems, web servers, and application servers.

distributing it to others) yields profits. In this model, the code itself – locked up and protected – matters most.

In the open source world, at least as defined by the GNU General Public License (GPL),[3] IP continues to play a critical role, but it is a different kind of IP. Dubbed 'copyleft,' open source IP focuses on keeping code access open, instead of closed. And, unlike in the world of proprietary software, the code matters less than the coder – anyone can see the code, but not everyone can replicate the coder's influence on the community to which she contributes her code. By virtue of her contribution, she builds influence in her chosen code community, which influence translates into a new kind of IP: reputation property instead of intellectual property.

In this new world of open source, reputation property means as much or more than traditional intellectual property. If I employ the developers on a given project, I have a measure of control over the direction that the open source project will take. But even more importantly, the more developers I employ who work on, say, the PostgreSQL database project, the more likely it is that would-be customers will trust me to be able to support it. Once a company is thought of as the default support vendor for a given project, the harder it becomes to dislodge that vendor. This jives perfectly with other commodity businesses where brand, price, and service provide the only lock-in, a benevolent lock-in *that customers choose* rather than one that vendors impose.

In this way, the open source code creator exercises a form of control over her creation, which control translates into *her* (and only her) ability to charge a premium for the software. Again, no law prevents another vendor from co-opting a vendor's GPL'd code and competing against the original creator on service, brand, and price. But the actual (still short, admittedly) history of open source trumpets the fact that no material diversion from the "customers buy from the source of the code" rule has yet occurred. The source code creator retains the exclusive (by practice, not by law) ability to charge a premium over commodity pricing.

As an open source creator, then, my options for deriving profit from my creation are not more limited, but they *are* different. Instead of a limited monopoly guarded by law, I have a monopoly guarded by common sense: buyers want to buy from the most qualified source of support. They pay to have access to the source: not source code, but source of the code.

This distinction is important. The importance of source code gets trumpeted so often that one would think that every IT buyer on the planet is clamoring for access to source code. They are not. Indeed, Microsoft recently conducted a survey of its customers and found that roughly 60% felt that access to source code was "critical." But when pressed on the matter, 95% said that they would never look at the source, and a whopping 99% said that they would never modify

---

[3] The GPL is but one of many different open source licenses, and is generally characterized as one of the most restrictive, if not *the* most restrictive. I use it here almost exclusively because its use as a strategic weapon is, in my estimation, more potent than less restrictive licenses like BSD-style licenses. Because it requires that all modifications to GPL'd code be GPL'd, as well, it is the best example of true copyleft in the open source licensing menu.

it.[4] (If they did, chances are that such modification would violate their support agreement, anyway, whether their vendor was Microsoft, Novell-SuSE, Oracle, Red Hat, etc.) In sum, customers perceive source code access to be important, but are not exactly sure why. As will be detailed below, the "why" relates to a desire for additional choice and control, which choice and control drive down costs.

Of course, nothing in this article should lead one to believe that source code is irrelevant. Source matters. It matters because it lowers switching costs (i.e., the cost of swapping out one vendor's software for a competing vendor's software); because it provides buyers with more control over their IT, as it allows them to shape code to fit their particular needs; and because it provides a mechanism for keeping vendors honest, by forcing them to take responsibility for the quality of their code. In short, source code matters because it shifts control back to the buyer, which forces vendors to offer better software at lower prices. While none of these source code benefits requires the intervention of a vendor, we should not get sucked into the belief that vendors matter but little in the open source world. Instead, open source actually makes vendors more relevant to customers than ever before at dramatically lower prices.

Besides benefiting customers, this GPL licensing scheme offers vendors a way to exercise an incredible amount of control over competitors. By open sourcing my code under the GPL, I push my competitors to follow suit or to increase their R&D efforts to escape commodification. Unfortunately for them, this counter-strategy of unilateral R&D arms race tends toward paltry results: customers will often opt for the "good enough" product when the price is dramatically lower. Yes, my closed-source competitors could simply take my freely accessible source code, "fork" it, and build it into their own products, but they almost never will. Doing so compels them to open source their own software, which they will be disinclined to do. Even if they did so, however, and even if my competitor is not a stodgy old closed-source vendor, but rather an agile open source predator, it would matter little, because open source buyers invariably favor the source of the source, as it were: they trust the creator of the code to support it best.

Open Distribution, Not Source

This has huge implications for the software industry. Disruptive vendors can opt to completely open source their code, relying on reputation property to net them revenues, and further relying on their freely available alternative to competitive products to force competitors to meet them on their home turf. This is not to say that all vendors must adopt an open source strategy, but rather that they must compete with open source's lower cost structures and superior distribution mechanisms. All must increasingly compete on open source's terms. More detail is needed on why this is so.

The Open Source Weapon

[4] So far as I know, this study has not been published. Jason Matusow, Microsoft's Shared Source Manager, detailed the results of the survey at the KMDI Open Source Conference in Toronto, Canada, in May 2004.

Open source enables a vendor to maximize its market penetration at minimal cost, which is the goal of every IT vendor, but particularly of emerging growth vendors seeking to displace incumbent vendors. One of the biggest roadblocks to any company's growth is the Bureaucracy Bottleneck – the larger the buyer (and, hence, the larger the opportunity), the more layers of bureaucracy an IT buyer must fight through to try-before-they-buy. Not so with open source, which surreptitiously makes its way into enterprises via free download.

Such distribution fattens a vendor's bottom line without fattening the customer's price tag. MySQL had 10 million downloads in 2003, and now in mid-2004 has over 5 million installations. Of these would-be customers, 5,000 have returned to buy a support contract/license from MySQL, bumping their revenues by 100% to $10 million in 2003.[5] The revenue growth is important, but even more so is that they achieved this growth by spending less than 10% of total revenues on sales and marketing activities. By contrast, most public software companies spend anywhere from 45% to 50% of total revenues on sales and marketing, and companies of MySQL's size generally spend 21.8% on these activities, according to a study done by Softletter.com.[6]. In short, open source creates a small universe of pre-qualified buyers who seek out the vendor, instead of the other way around, with the vendor's primary marketing costs relating to setting up an FTP server and mostly word-of-mouth-type evangelism to developers.

The savings do not stop there. Whether the open source vendor "borrows" much of its code (e.g., Novell, Specifix, Gluecode) or creates it almost entirely in-house and then open sources it (MySQL, SugarCRM, JBoss), open source delivers development-related cost savings. For the "borrowers," the cost savings are obvious: they leverage a well-developed body of code, most of it written by individuals not on their payroll. For the JBosses and MySQLs of the world, who do 85% to 100% of their own development work, they still derive significant QA savings from the global pool of testers who submit bug fixes and code contributions (which may or may not be used by the vendor). Cheaper to build, cheaper to sell, cheaper to buy.

Proliferating Open Source Beyond the Enterprise

Today, open source largely confines itself to infrastructure software, in part because this is where the widest computing community resides. Community-based open source projects require a sufficient body of developers with aptitude and interest in a given development problem. These constraints have bounded the range and volume of successful open source projects. As Joel West (San Jose State University) and Siobhan O'Mahony (Harvard Business School) point out in a June 2004 research paper, Linux and Apache are anomalies – most open source projects are either half-baked or non-existent. Take a walk through SourceForge.net today and you will find that of the roughly 46,000 projects hosted there, 59.38% are pre-beta and only 1.57% were classified as mature. 25% of the projects listed as of 2003 did not even have software in their

---

[5] Sources for this data include MySQL (conversations with its CEO, Marten Mickos, as well as its Vice President of Marketing, Zack Urlocher) and an exceptional business case study from Stanford Graduate School of Business, authored by Christof Wittig, currently CEO of db4objects, and Sami Inkinen, a fellow Sloan student at the GSB. The case study can be sourced as follows: *MySQL Open Source Database in 2004*. Case: SM-124, June 1, 2004.

[6] *Softletter Financial Handbook*, pages 10-11. Accessed September 1, 2004; see www.softletter.com/pdf_files/FHp10-p13.pdf.

code repositories. Clearly, while volunteer labor has managed to create some exceptional projects (Linux, Apache, Samba, PostgreSQL), these are the exception.[7]

This is not to say that open source is weak as a development methodology, nor that it will wither and die. Rather, it simply means that the open source community, in its purest sense, will generate fewer of the industry's open source projects over time as corporate-sponsored projects flower and bloom. As the IT industry begins to recognize the promise of emerging open source business models, the open source community will drive relatively fewer successful open source projects. As this happens, no area of the software "stack" will be exempt from open source's influence and intrusion.

The open source communitySignificantly, open source business models will pave the way for open source to conquer the Great Middle Class of IT: the small to medium-sized enterprise (SME) market. Open source, as a disruptive methodology in the Clayton Christensen sense, will do more than simply allow startup vendors to compete against established vendors in established markets: it will help to create new markets by competing against under-consumption and non-consumption. The Internet, and open source's unparalleled use of it, is at this trend's core.

In *Does IT Matter?*, Nick Carr offers one example of how this worked in another industry: chocolate.[8] Milton Hershey, founder of Hershey Chocolate, noticed a gap in the chocolate landscape, a gap that the railroad could resolve. Until his observation, transportation deficiencies had forced production to stick close to consumption – there was no quick and refrigerated way to ship chocolate over long distances, creating scores of micro-markets for chocolate. With the advent of the railroad, however, Hershey conceived the idea of a national chocolate market, built it, and owned it.

Today, the Internet parallels the railroad infrastructure of Hershey's era. It offers software vendors (ISVs) a similar opportunity to that which Hershey had, yet the majority of ISVs are not capitalizing on it. Yes, some traditional ISVs can and do offer their products for download, but this is a makeshift attempt to leverage the Internet. *Open source is an Internet phenomenon* – it depends upon the Internet and extends the Internet's utility. Open source should disproportionately benefit from the Internet's distribution mechanism, provided companies understand this fact and act accordingly. As will be shown below, tomorrow's most successful software vendors will triumph to the extent that they develop models that leverage the Internet as a distribution mechanism, and use open source licensing as the rules-based system to govern that distribution.

So, Why Not Freeware?

Let us assume all of this is true. Open source is great because it enables upstart competitors to undercut established vendors on price while providing their customers first-class technology at economy-class pricing. But if open source is so important because it allows me to freely

---

[7] Joel West and Siobhan O'Mahony, *Contrasting Community Building in Sponsored and Community Founded Open Source Projects*, Free and Open Source Research Community (Online). June 23, 2004; see http://opensource.mit.edu/papers/westomahony.pdf.

[8] Nicholas G. Carr, *Does IT Matter*?, Boston: Harvard Business School Press, 2004.

distribute my product over the Internet, why is freeware not equally disruptive?[9] Stated another way, if the source code does not matter, and only distribution matters, then why not just give the software away as freeware and charge users who require support? Why offer something (access to source code) that simply does not matter?

The easy way out of this apparent quandary is to allow that while open distribution matters most, open source code access is also important. But this does not get us very far. It is therefore important to detail the reasons that freeware cannot match open source as a distribution strategy. Most importantly, it needs to be shown that the two matter most when they intersect, making distribution without access as hollow as access without distribution.

Don't View. Don't Modify. What Do You Do?

As mentioned above, it is an indisputable fact that the vast majority of IT buyers will never view or modify source code, even if offered the ability. There are numerous reasons for this, but the most compelling one is that customers expect to pay for a solution to their problems, and not merely a tool to help them solve their own problems. (More on this below.) No company can afford the time and human resources necessary to resolve all IT problems; therefore, they take "shortcuts" by buying software that purports to fix certain problems for them. This applies equally well to closed source software and open source software. Most IT buyers just want their software to work, not to fiddle with it.

By opting not to view or modify source code, does an IT buyer thereby opt out of any and all of the benefits of access to the source code? Absolutely not. Just because customers do not choose to exercise their rights to view and modify source code does not mean they do not benefit from the right, even when not exercised. On one level, the option to view the source code serves as a surrogate for the actual exercise of this ability. As an example, because I can review the database code that Sleepycat delivers to me, it forces Sleepycat to provide a higher-quality product than closed source vendors would have to offer.

R0ml Lefkowitz, Chief Technical Architect for AT&T Wireless, gives a tangible example of how this works. In June 2003, R0ml related that he had asked his wife to solicit multiple contractors' bids for a home improvement project they intended to begin. Instead of gathering several bids, however, R0ml's wife chose to procure only one bid. When he asked her why, she responded that she figured the contractor would assume she had collected a number of bids, and so would give her his best bid from the start. The option to exercise choice, then, served her as a useful surrogate for actual choice.

In this way, access to source code motivates the code's vendor to provide a superior product, knowing that it will be open for all to see. It also functions as a security blanket for customers. Hopefully, they will never have to look at the source code. But if Vendor X fails to deliver on its promises, or if it goes out of business, that customer will have the option (unpleasant though it might be) to have some other services firm support the stranded code. Source code access lets buyers rely on their vendors…but not too much. Importantly, the more

[9] Larry Augustin, Venture Partner with Azure Capital and Chairman of VA Software (formerly, VA Linux), originally needled me with this question, for which I thank him.

independent the buyer is from the vendor, the lower the vendor's prices must be. More on this below.

As IT buyers have grown comfortable with open source projects, another benefit has emerged. Initially, it is true that buyers will tend to want to avoid tampering with the software they buy. Over time, however, as they grow familiar with a product (closed or open), the buyer's developers will want to make tweaks. They begin to support themselves, in other words, because calling out for support takes unnecessary time (and patience). In a closed source world, however, their ability to tweak the "solutions" they buy is limited. In an open source world, the Amazons of the world have free reign to rein in their IT and exercise control over it. Source code access gives customers the ability to experiment and tailor software to their liking, if they choose and on their own timetable.

Other reasons have been suggested. Developers like to be creative. Like anyone else, they prefer not to have self-expression manacled, and they choose to express themselves in the code they write and modify. On the fsb@crynwr.com mailing list in June 2003, Frank Hecker, a systems engineer with Opsware, argued that access to source code is critical for developers at the heart of a company's IT infrastructure. Such people want to be able to modify code, and so must have access to source, because they will need to be able to fix any problems they may download into the company. Outside the corporate firewall, and also on the fsb@crynwr.com mailing list, Ben Tilly, a noted Perl hacker, stipulated that while the majority of developers are not involved in open source, access to source code is critical to that core of developers who do participate. They are the lifeblood of open source communities, and are the ones who will openly extend assistance to newbies who just want the code to work without getting their hands dirty.

All of which is a verbose way of repeating the earlier point: source matters, even where it may not directly matter to the end user. Access to source extends benefits to users beyond those chosen few who actually exercise the right to touch source code. Source matters because choice matters. Choice matters for a number of reasons, not least of which is that choice drives down prices. And choice is amplified by open source, not by freeware, which has its source code closed.

Open Source. Open Choice. Open Wallet.

Many successful software vendors would have us believe otherwise. That is, they want to sell suites of services that take care of all needs; that reduce complexity; that reduce choice. The primary perpetrator of this strategy is Microsoft which, as Dana Gardner of Yankee Group notes, wants to "make the end user any offer they can't refuse -- to go Windows everywhere."[10] One major problem with buying into these monolithic visions is that once in, the switching costs to go with another vendor are prohibitive.

By buying into Microsoft or any other vendor that holds out greatly reduced choice as a way to accomplish moderately reduced complexity, a buyer surrenders its IT destiny to that vendor.

---

[10] Qtd. in Vincent Ryan, "Behind the Scenes: Microsoft's Server Dominance," *Enterprise Windows IT.com*, August 22, 2003; see <http://enterprise-windows-it.newsfactor.com/story.xhtml?story_id=22143>

It upgrades when the vendor wants it to. It gets new technology when the vendor chooses to innovate. (I would argue, and have on several occasions, that Microsoft's market dominance has caused it to stagnate in terms of innovation. When was the last time that Microsoft's Office product significantly improved over the last version? And yet buyers keep buying, because they find themselves on the Microsoft treadmill.) And it pays whatever the vendor demands, because the customer has no other options. It is a prisoner of the vendor's universe, however expansive the vendor pretends it to be.

Over time, buyers who condemn themselves to such vendor-controlled realities will pay more for their IT, both in hard costs and opportunity costs. Open source offers the opposite vision: maximum freedom to shift between vendors (even while staying with the same or similar code base). Open source therefore costs less, short-term and especially long-term.

If we step outside the IT world for a moment, this point will become even clearer. My wife and I recently redid our landscaping, including our cement work. Or, rather, we wisely chose to hire out the work. True, with a Dummy's Guide to Cement (my "source code," as it were), even I might have been able to figure it out and could have completed the project satisfactorily. Had I opted to do the work myself, the cost would have been X. Because I could have done the work myself for X, my cement contractor was only able to charge me 1.5X. Had he bid higher, I would have had strong incentive to perform the work myself. I had access to the source – his pricing power was thereby curtailed. (In the same way, access to a closed binary, as with freeware, does not accomplish this same effect of driving costs down.)

The cement contractor ended up performing shoddy work, and walked off with a portion of our money, for which he had not completed the associated work. Measuring it out, it will cost us 5-10X to hire a lawyer to compel our cement contractor to satisfactorily complete his 1.5X worth of work. I happen to be a lawyer, but not one that has ever actually practiced law, so I am stuck paying the lawyer's fees. $500/hour to recover $1500+ in payments owed to me by the contractor.

Perverse world, you say? Yes, I suppose so, but the point is that the delta between the cost of me doing my own cement work and the cost of me doing my own legal work is directly proportionate to the skill set involved and the artificial licenses set up by the legal profession to keep would-be attorneys in "would-be land." I am effectively barred from accessing the "source code" of the legal (and medical, among others) profession, which drives up the price that I must pay.

Again, access to source code, whether in software or cement, offers choice, and choice ensures lower prices. It does not matter that most people will never choose to do their own cement work, just as it does not matter that most IT buyers will never choose to view and modify source code. The important thing is that they could if they were so inclined. That "could" is instrumental in dropping prices through the floor.

Such lower prices then allow the CIO to spend more money on developers who can further customize software to meet that specific organization's requirements. Imagine that: IT that works for a customer, rather than against it. *That* is innovative.

Such innovation is what open source is all about, and is why it continues to make inroads in the enterprise, in embedded devices, and everywhere else. Open source brings choice, and choice saves money. Freeware does not engender such choice. Meaningful choice is not created

by cost-free technology; rather, choice is created by the freedom to manipulate code to one's personal (or corporate advantage), or to have it widely distributed in such a way that one can benefit from others' exercise of that freedom. Access matters little without distribution, and distribution matters little without access. Together, they spell the commencement of a new age of software innovation, innovation that benefits vendors' and buyers' bottom lines.

Open Source Business Models

All of this may sound plausible enough, but we now need to trace through some real-world business models that vendors use or could use to leverage the benefits of open source to drive revenues and boost profits. Before doing so, it is important to note that an "open source business model" means more than simply supporting Linux as an operating system. In other words, the fact that my CRM system runs on Linux, or that my hardware appliance has Linux as its core, does not make me an open source company. An open source business model means that a vendor somehow engages with the open source community.

With that said, I also need to stress that whatever the importance of open source, not all companies must adopt open source to find success. Open source is the great commodifier, but there will always be those who successfully evade that commodification. Other industries prove instructive on this point.

Take retail, for example. Even as low-cost commodifiers devour middle ground in this market, profits persist "up the stack." Wal-Mart is the 8000-lbs. gorilla of commodification, cannibalizing groceries, clothing, and just about everything else to which it puts its hands.[11] Still, for all Wal-Mart's success in low-end fashion, for example, Nordstrom continues to win at the higher-end game. This is more than a case of customer snobbery – it has to do with an experience that Nordstrom delivers (superior customer service) that Wal-Mart is structurally incapable of offering. (This same phenomenon exists in coffee – why are consumers so willing to shell out $4 for a cup of coffee? Because Starbucks has defined a customer experience that transcends Maxwell House at home.)

Another example is the groceries market, as Charles Fishman highlights.[12] In just a few short years, Wal-Mart has become the United States' largest grocery chain, yet the title of "Most Profitable Grocery Chain" and "Fastest Growing Grocery Chain" eludes Wal-Mart (or its European competitor, Aldi). No, those titles go to Whole Foods, with remarkable year-over-year growth in the face of a nationwide 2.5 annual compound growth rate: 17% (2003), 21% (2002), 21% (2001), 23% (2000), and 14% (1999). Whole Foods delivers an upscale grocery experience, offering organic foods and superb quality, while Wal-Mart stocks its shelves according to its *modus operandi*: decent selection at rock-bottom prices. Both chains have found their respective strategies to be highly profitable – everyone else has gobbled their dust. Whole Foods has

---

[11] For those outside the US who are unfamiliar with Wal-Mart, not to worry. Wal-Mart will be on your doorstep (and probably already is – they own significant properties throughout Europe) in the very near future.

[12] Fishman, Charles; "The Anarchist's Cookbook," *Fast Company*, Issue 84, Page 70 (July 2004); see http://www.fastcompany.com/magazine/84/wholefoods.html.

registered $188 million in profits over the last several years, while Food Lion cleared only $150 million with seven times as many stores and five times Whole Foods' revenues. Safeway, for its part, lost $1 billion in the same period on even greater revenues. The takeaway? There is no room in the middle for undifferentiated players. One either commodifies or evades commodification through innovation. Everyone else languishes.

Both Source (aka "Mixed Source")

So, the first business model is for the technical innovator that refuses to join open source commodification at all. But what about those companies that opt for a 'both source' model, whereby they offer both open source and proprietary software? This model has promise and peril, requiring the vendor to walk a fine line between the model's divergent business requirements (low-end commodification/standardization coupled with high-end specialization). To the extent that a company marries the two, it must do so with a clear understanding of open source complements and substitutes to its proprietary product portfolio.

Both Source offers a way to fill in the gaps left by open source, and charge a premium for this 'service,' while still delivering open source software. Such a model seems to be ideal for established players who cannot abandon existing customers of closed source products, and blanch at the thought of losing existing profit margins. Of course, whether a vendor can avert the open source "threat" depends upon whether or not open source has created a viable *substitute* to its product. If so, head-on competition with that open source project is likely futile unless it can move significantly upward in the feature set. Even if it can, competing against free and 'good enough' is exceptionally difficult.

A Both Source strategy makes more sense where the vendor can define and contribute to open source complements. In economic terms, a complement is something that completes a whole; in software terms, it is a component of a software solution. So, just as French fries may be considered a complement to a hamburger, so, too, is Apache a complement to IBM's Websphere product. Importantly, the more complements that exist for a given product, the more desirable that product becomes for customers, so vendors want as many low cost, high quality complements for their product as possible. Oracle likes Linux and x86 hardware because it drives down the total cost of a customer's database solution…without lowering the cost of Oracle's software. Customers, thus, can buy more Oracle software, which gives Larry Ellison more time on his boat.

So far, this sounds like a reasonable defensive strategy for vendors that want to toe-dip into the open source community without getting very wet. But Both Source also allows vendors to take a scorched earth agenda against their competitors, by skillfully choosing to build open source complements to their proprietary software, complements which cut directly at those areas that competitors have chosen to retain as proprietary. Of course, such a strategy must bear in mind the distinct possibility that the opposing vendor will then choose to open source pieces of its portfolio that injure the first vendor, creating a lot more open source software, but not necessarily any profits for either company.

Still, it is an open question whether this is a solid strategy against upstart competitors with lower costs who can undercut a proprietary product's margins. In addition, a Both Source strategy works best for vendors with products that have respectable market share. Slapping

open source on or around an also-ran product will not sell it. Good technology, good service, and good sales/marketing sell products. Open source, by itself, is as much a losing strategy as closed source, by itself.

Open source complements to a market leader's products make that proprietary product more valuable by lowering the total cost of the product. Hence, while Both Source may be the easiest step for a closed source company to make, it will only help the vendor if its products were already competing well against other proprietary products. Again, Both Source offers no panacea for market losers. The lesson? Companies should adopt the Both Source strategy when they are on top of their game, rather than when they are losing the final set of their match. For market losers, a better bet is to make the difficult transition into a pure-play open source vendor, as defined below.

Professional Open Source (aka Services Model)

The dirty little secret of open source is that "open source community" is something of a misnomer. In general, the actual number of contributors to any given project, including the Linux kernel, is tiny. Thus, to "own" an open source project requires little outlay of human resources in terms of numbers, though it may require a significant amount of time to build reputation capital within a given open source community. (Newbies to the Linux kernel, for example, should expect to put in two years or more before they can hope to attain "committer" status in the kernel hierarchy.) Despite the low number of developers required to corner the market on an open source project, the importance of doing so is massive: employing a majority of the developers on a given project roughly equates to intellectual property ownership, as explained above.

For this reason, companies that spawn open source projects, e.g., JBoss, are able to completely open source their code without abandoning pricing power. JBoss, for its part, employs roughly 85% of the developers who contribute to the JBoss open source project. JBoss offers their code under the LGPL (Lesser General Public License), which allows users a wide range of action vis-à-vis their code, including the right to fork the JBoss project and start JBoss II.

But no one does that, for reasons already detailed.

Because of the heavy JBoss "ownership" of the committers to the project, the company does not save a great deal of money on development costs. They function much like any closed-source company, except that their development is open for public view and consumption. Any appreciable development savings derive from the bug finds/fixes that JBoss receives from its development community.

Still, the Professional Open Source business model is not really about development savings. Rather, it is about maximizing distribution of one's product; getting it beyond the Purchasing Firewall/Bureaucracy Bottleneck to plant the product in the hands of its developer end-users so that they can try and then revisit the Professional Open Source vendor for support/service contracts. To get approval to use BEA's Weblogic or IBM's Websphere, a developer would need to go through a cumbersome process. To use JBoss, she simply needs to click 'Click here to download.' And while the developer might choose to support herself through newsgroups or other online fora, in production situations she will generally turn to the source of the code (in

this case, JBoss). This is the classic open source model, though it is only now starting to be exploited effectively.

Dual-License Model

The Dual-License Model has been popularized by MySQL, but has been around for some time, most notably deployed by Sleepycat and Trolltech. In the case of a dual-license vendor, that vendor employs not most but *all* of the developers who contribute to the code. Because they employ all of the developers, they also own all of the copyrights to their work. Then, as the owner of the copyrights, they are entitled to license their software under one or more different licenses.

However, the fact that they own the copyrights and employ the developers begs the question as to what benefit, if any, they derive from their open source status. The answer, as with the Professional Open Source model, lies in distribution strategy. For a dual-license vendor, open source is less a matter of development and more a matter of distribution for open source vendors. Yes, they derive benefit from outside developers who contribute code (though MySQL, for example, tends to repurpose/rewrite incoming code to help it better fit their existing code base) and bug fixes, but their primary benefit is in the ability to broadcast their product to the world with customers benefiting from lower prices and less lock-in.

Also interesting, though not a benefit touted by the primary adopters of the dual-license model (MySQL, Sleepycat, Trolltech, and now SugarCRM) is the fact that the dual-license strategy provides customers with a mechanism to buy their way out of the GPL, if they feel this desirable. This is of particular benefit in the embedded world, where Linksys, for example, might receive GPL'd code from Broadcom and might want a closed-source license to that code, such that they will not have to open source the software running their routers and access points. (A purely hypothetical example, of course….) db4objects is promoting its embedded database with precisely this message, one that customers appreciate because however much a vendor may prefer the GPL or another open source license, the fact remains that it may not always be the best fit for a given customer. As such, the Dual-License Model offers customers a way to pay for the right to choose the license under which they receive their software.

ASP Model

Such are the prominent open source models that are easily recognized as such: open source business models deployed by open source companies. But, as Tim O'Reilly has been telling the industry for years, "open source" is a much bigger tent than we may recognize. Tim includes such "infoware" vendors like Google and Amazon in his open source tent. The common denominator between the two? Internet infrastructure powered by open source (Linux and a great deal else), plus an architecture that promotes participation, which participation makes the infoware increasingly valuable.

The enterprise IT industry has also been moving toward a related model for standard enterprise applications, calling it utility computing, on-demand computing, and a range of other names. In this model, IT vendors deliver computing power in a utility fashion: Enterprise Consumer X gets the computing cycles when it needs them, rather than buying all of the hardware/software upfront.

Importantly, customers in this model buy IT (including software) as a service, rather than as a standalone product. As such, customers do not really buy software at all - they buy a solution to their business problem. Whether the "guts" of that solution are open or closed source does not matter anymore. Customers simply pay for value, delivered as a service: SP (service property) rather than IP (intellectual property).

This sounds much like software's ASP model, in which software is delivered to the customer over the Internet, hosted on a central server by the vendor, with customers paying for the value they access over the network. A prominent example is Salesforce.com. Whether the software underpinning the service is closed or open source becomes irrelevant. The requirement that I release my modifications to a given open source project is triggered by distribution – so long as I do not actually distribute the code (but merely the resultant service dictated by the code), I need not open source my modifications, unless I choose to do so.

Other Models

The business models above are the primary models in use today by most open source companies. However, some of the most interesting new companies employ equally interesting (and innovative) business models, generally altering the way open source software is supported. For such companies, the real customer benefit of open source is the availability of source code. But, as noted, major vendors like Novell and Red Hat, who tie their support contracts to specific product builds, obviate this benefit. Gluecode and Specifix resolve this irony and may point to tomorrow's most successful open source business models.

Managed Source Model

Gluecode incorporates three levels of source code support into its model. First, Gluecode conglomerates various Apache packages, tests them, and generally makes them play nicely together so that customers need not worry about visiting Apache.com for themselves. Second, Gluecode develops its own proprietary software to extend Apache's reach and thereby provide a compelling solution for portals/business process management (BPM). Third, and unique to Gluecode, the company offers source-level support to its customers, allowing them to check-in their code to Gluecode's CVS repository. Gluecode runs the customer's code against test suites to ensure the customer's modifications/additions will work properly with Gluecode's open + closed code base, enabling the customer to become something more: a development partner.

Code-Level Service Model

Specifix does something similar. Focusing on embedded and server deployments of Linux, Specifix allows its customers to modify Specifix's Linux distribution to meet their particular requirements. Instead of invalidating their support agreement with Specifix by doing so, Specifix tracks exactly where the modifications were made and allows the customer to support its own modifications, while continuing to support the original Specifix distribution. The customer may, in other words, opt for the "road less traveled by," but Specifix is happy to maintain the more-trafficked road for them, keeping it parallel with the customer's chosen divergence.

Conclusion

Open source propels software toward Commodity Land, a happy place where customers pay for real value and vendors compete on that value, not intellectual property lock-in. Each of the open source business models detailed above will help to further this trend, making open source mainstream and possibly displacing the traditional, IP-based model as the default.

We are thus on the cusp of a Kuhnian paradigm shift,[13] one that will fundamentally alter the way IT vendors create, sell, and distribute software. Once apparently stymied by the restrictions that open source licensing places on traditional business practices by IT vendors, open source vendors are now finding that open source licensing creates as many opportunities as it closes, changing the nature of software competition for decades to come. This means that incumbent and emerging IT vendors must understand the new rules of engagement to compete effectively. Whether they like to admit it or not, open source will force every software vendor to come to grips with omnipresent, ravenous commodification.

Some may opt for technical innovation over business model innovation, with varying degrees of success. However, such innovators should recognize that while copyright and patent provide potent protections, they also put the vendor in an adversarial relationship with the customer. As such, these traditional intellectual property tools hurt customers as much (or more) than they do competitors, and will put them at a disadvantage against open source competitors who offer customers choice and value at lower prices. Open source, then, allows vendors to lay waste to their competitors' profit margins by lowering their own costs of distribution, sales, marketing, and development, while simultaneously blessing their customers with increased IT flexibility and a more finely tailored approach to solving their business problems.

Open source, then, offers a new way to innovate, a new way to compete, and a new way to win.

Copyright © Matthew N. Asay

Biography

Matt Asay is director and founding member of Novell's Linux Business Office, where he is responsible for helping to chart and implement the company's open source strategy. Asay is also an Entrepreneur-in-Residence for Thomas Weisel Venture Partners, a $250-million venture fund

---

[13] Thomas Kuhn, one of the world's foremost philosophers of science, coined the term, "paradigm shift," in his classic book, *The Structure of Scientific Revolutions*. As Kuhn noted, science tends to progress incrementally, until someone (e.g., Copernicus) sees the world in a dramatically different light, disrupting science onto a new path of discovery. In short, scientists tend to "do" science within their narrowly-defined, self-imposed paradigms until someone comes along and is able to see beyond the traditional view, a bit like Plato's philosopher-king seeing beyond the shadows of the cave to grasp the magnificence of the sun. Tim O'Reilly has written an excellent piece on how Kuhn's thought applies to open source; see http://tim.oreilly.com/opensource/paradigmshift_0504.html.

based in Silicon Valley, where he focuses on open source-related investments. Asay founded and continues as managing director of the Open Source Business Conference (OSBC), the industry's preeminent open source strategy conference.

Prior to joining Novell, Asay was General Manager of Lineo's Network & Communications division. (Motorola acquired Lineo, an embedded Linux startup, in 2002.) He earned his Juris Doctorate degree from Stanford Law School, where he spent two of his three years working on open source licensing issues with Larry Lessig and other Stanford faculty. Asay also holds an MA from the University of Kent (Canterbury, UK) and a BA from Brigham Young University.