14

Creativity and Control

"Practically speaking, copyright and patent and trademark law have only one thing in common: Each is legitimate only as far as it serves the public interest. Your interest in your freedom is a part of the public interest that must be served."
—Richard Stallman,[1] founder of the GNU Project

In the previous two chapters, we took a pragmatic approach to intellectual property (IP): Understand how the existing system of protections works, and then leverage that system to create a range of techno-business models, from narrowly controlled (proprietary) to globally participative (open). We hoped to show the range of possibilities without being overtly biased about recommending a particular path. And we avoided posing basic questions such as whether the existing system of protections is best for fostering innovation.

In this chapter, we will definitely let our biases percolate through. If we are successful, you should leave this chapter understanding our fundamental belief that in this new era, innovation is best served by structures that promote openness and sharing—that there is a dynamic tension between creativity and control, and that by easing back on the controls, creativity will become even richer.

This viewpoint is, of course, not original to us. We've been very much influenced by the writings of Lawrence Lessig,[2] who in turn has stood on the shoulders of pioneers such as Richard Stallman, Eric Raymond, Mitch Kapor, Eben Moglen, and Linus Torvalds.

Maximizing the Cycle of Innovation

The instruments of control—from patents, copyrights, and trademarks, to digital rights management (DRM) technologies—seem to be at odds with creativity. After all, these controls are all about *excluding* or *restricting* others from using an idea, or copying works and content. Yet very few creative works,

including software programs, are truly original. We invariably build upon the works of others. As filmmaker Martin Scorsese said: "The greater truth is that everything—every painting, every movie, every play, every song—comes out of something that precedes it.... It's endlessly old and endlessly new at the same time."

Put another way, most creative works are in fact derivative ones. They incorporate ideas (and their expressions) from others and then build, reinterpret, improve, optimize, and ultimately return better ideas and better-engineered artifacts. If the instruments of control are set right, this virtuous cycle continues, and a rising tide of innovation lifts all boats.

Moore's law is a superb example of the innovation cycle—so much so that we can accurately predict that a key innovation metric of semiconductors (transistors per chip) will indeed double about every eighteen months. This is not a law of physics; it is a law of techno-economics. Fifteen years ago, for example, very few people could predict exactly how we would stay on this exponential cycle, and yet ten doubling times later we have about a thousand times (!) more transistors per chip.

Evidently, our system of intellectual property controls in the business of semiconductor manufacturing is having some positive effects. Companies all along the value chain, from equipment suppliers to fabricators, continue the enormous cycles of R&D and capital investment that manifest Moore's law. And there are similarly breathtaking capital cycles in the pharmaceuticals industry, with $1 billion spent on bringing each new "molecular entity" (drug) to market, meaning tens of billions of dollars are spent in R&D each year.[3] Intellectual property controls play an essential role in this industry to exclude others from the much easier task of replicating a compound that took many years of forward investment to discover and test.

A trap to avoid is generalizing from these long-term, very capital-intensive innovation cycles, and concluding that everything is working as it should. The question we should always be asking ourselves is whether we have struck the right balance:

"Does our system of controls maximize innovation fairly?"

The *fairly* qualifier deserves some explanation. By this we mean fairness in both access to markets by competitors along with fairness in compensation for one's creative efforts. Overcontrol can damage both senses.

Our view is that our system of controls is far from the best one. Like Lessig, we believe a better balance can and should be struck along the control-creativity continuum that serves the interests of all: engineers, businesses, and the public. We view Free and Open Source Software (FOSS) as the harbinger of systems that unleash very rapidly moving innovation cycles through sharing and global participation.

To be sure, extraordinary FOSS innovation has happened under the current system of intellectual property controls. The genius of Stallman and Moglen with the Free Software Foundation (FSF) and Software Freedom Law Center, and Lessig with the Creative Commons, has been to parlay the existing copyright laws into a whole new set of concepts regarding the licensing of IP. As described in Chapter 13, this is accomplished by writing a new license (e.g., an "open source" one) that is incorporated into copyrighted material. If you don't follow the terms of the license, you run afoul of the existing copyright law.

Perhaps our existing system is malleable enough to get us to a better balance. Perhaps. We think there are many opportunities for reform, from the issues of software patents and patent trolling to the absolutism of control over interfaces and DRM.

But no matter your view on the necessity of reform, there are plenty of ways you can be progressive and tap into the tremendous power of creative, freely moving global innovation networks.

How We Got Here

In Chapter 12, we provided a primer on the primary landscape of intellectual property law as it exists today: patents, copyrights, trademarks, and trade secrets. Here, we'll provide a bit of context as to why we ended up with our current system as a segue to the discussion of what could be next.

First, let's back up and look at the term *intellectual property*. It's a loaded one. Many people—the authors of this book included—bristle at it because it evokes the idea of real property, which it clearly isn't. We also surround the term with analogues of real property: People "steal" software or music; they become "pirates." Promoters of participation and sharing get labeled as "communists" (we assure you, the book's authors are most definitely capitalists).

Why is IP unlike real property? Because the consequences of its misappropriation can be wildly different. Think of an apple (the fruit). The apple is real property. If I give or sell my apple to you or you steal it from me, you have one and I'm deprived of one in a very direct and measurable way. Same thing if you steal my vinyl record of *Led Zeppelin IV*. However, if you "steal" a copy of my digital music, I am not deprived of anything other than the money you might have paid, and *I still have my music*.

And the damages for IP infringement are often incongruent. If you dump toxic sludge on my front lawn, you have to pay something proportional to what the real property damage was (oh, and maybe something punitive on top). When you get punished for patent infringement, the settlements can be

surreal. A patent holder, for example, can extract hundreds of millions of dollars for a patent he may have paid thousands for, and whose infringement causes him no direct harm whatsoever.

With our discomfort with the term *intellectual property* duly noted, let's not get bogged down over it. Protection mechanisms for IP seem to have always had a role in human cultures. Look at any era and you'll find examples. In ancient Greece, inventors of new recipes were granted an exclusive right to make that food for one year. The Republic of Venice started issuing patents in 1474, protecting inventors by prohibiting others from replicating their new devices.[4] The United States has had a copyright law since 1787, and in 1884 the U.S. Supreme Court concluded that photographs could be copyrighted.

Moreover, intellectual property law has long played a role in shaping the development of key innovations—sometimes encouraging new ideas, sometimes inhibiting new ideas, sometimes encouraging new ideas by getting in the way. Here are some oft-cited examples from the history of patents in America.

- In 1781, James Watt, who built the first full-scale steam engine, was forced to circumvent a patent held by James Pickard in order to make his engine's flywheel turn—and as a result his team invented the "sun and planet gear" which played an important part in the development of devices in the Industrial Revolution.[5]
- Between 1878 and 1892, the electric light industry was growing in terms of installed lights but shrinking in terms of competition as both Thomas Edison and George Westinghouse determined to control the industry and its advancement. They formed the Board of Patent Control, a joint arrangement between General Electric and the Westinghouse Company, to defend the patents of the two companies in litigation. This proved to be a wise decision as more than 600 lawsuits for patent infringement were filed.
- The development of radio technology was blocked by the patents held by various individuals and corporations, and it wasn't until the U.S. government stepped in and forced cooperation among various competing patent-holders that the technology moved forward.[6]

Students of the field (and the professionals who practice it) would all point out that these are examples where the patent system "maximizes innovation" by either sparking new ideas in an effort to avoid existing patents or encouraging the pooling of IP among competitors, thus advancing collective innovation.

Let's look at this latter aspect in more detail. Big patent exchanges or cross licenses among companies are often seen as signs of a healthy system. One of the most visible consortia is the MPEG LA[7] which is a clearinghouse licensing authority for a plethora of patents relating to the MPEG video and audio encoding formats (MPEG-2 and MPEG-4, in particular). If you wanted to, say, build a new set-top box that would plug into the Internet and play MPEG video streams on a TV, you'd be advised to obtain a license from MPEG LA. Expect to pay several dollars per box for the pleasure[8] which might be about the same amount of money in net profit you were planning on.

But here it is, all in one place. A set of intellectual property that enables the worldwide encoding and playback of digital media. That would seem to be an ideal outcome: an opportunity for great technology to be developed and standardized and a way for compensation to flow back to the inventors. Certainly such a system is best for innovation, isn't it? In our view it isn't. A better system is one that would enable a much freer flow of ideas and implementations.

From another vantage point, licensing the intellectual property around broadly adopted interoperability standards is mostly just a tax. But more importantly, the tax rate can be set arbitrarily, even much higher than the intrinsic value of the underlying innovation. The thing that makes a standard valuable is that *it is the standard*. In video encoding, for example, there are lots of alternatives, many of which are technically superior to the MPEG suite. So, the value of the MPEG suite is that we all agree upon it. And by agreeing upon it, we get to create a very strong network effect. That network effect—the consumer benefit of being able to view any video on any player—becomes incredibly more valuable than, say, a new algorithm that compresses at lower bandwidth with higher quality.

To have IP claims come in *ex post* (after the standard has been agreed to) is unfair at best and mostly just plain exploitive. At a minimum, we strongly believe that IP claims surrounding a potential standard should all be aired *ex ante* (before the standard is agreed to) and Reasonable and Non-Discriminatory (RAND) pricing exposed so that at least there can be an informed discussion about whether a particular bit of innovation is actually worth the license fees. We have little doubt that if we were to move to this level of transparency, many other less-expensive alternatives would surface.

Ultimately, we are convinced that royalty-free alternatives would emerge, and these would be strongly preferred. Why? Because in this case, the system becomes open to cycles of improvement that can come from anywhere. There is no monetary franchise that is being protected by keeping the world attached to a stagnant technology that generates income for the promoters of the standards in the first place. Furthermore, the downstream innovations

also open up because many more people can figure out ways to use the standardized technology in new and unanticipated ways.

None of this discussion around IP as related to standards should be construed as an argument that we should abandon the patent system. We are pointing out a particularly (in our minds, at least) anti-innovation consequence of when the intrinsic value of an invention is nowhere close to the network value that can be created, especially through the process of creating standards. It's an area in which we all should proceed very carefully.

"We live in a world now that consists of pipes and switches," said Dave Crossland in his legendary talk at ITC-ILO in Turin back in December 2004. "Pipes that move things from place to place, frictionlessly, at the speed of light. And switches that determine who gets which things, when, how, with what control, and at what price . . . and the rules that they use to determine who gets what, when, where, how, and at what price, are computer programs. Those who control computer programs, control who gets everything."

Control over Interfaces

The issues that arise concerning IP and standards are actually a special case of a bigger concept: controlling markets by controlling interfaces. By *interface* we mean an essential agreement (mechanical, electrical, logical) that connects things. A simple example is a razor handle and razor blade cartridge. The interface is the mechanical linkage between the two: the way the razor cartridge clicks into the handle.

The point we make here is that the interface is an essential feature to control. If, for example, you obtain a patent on that mechanism, you can effectively *exclude anyone else from making a compatible blade*. That is extraordinarily powerful. You can completely control the marketplace for one thing (the blade) by controlling the way it connects to another (the handle). And this supports a key business model; give away the handle and charge (handsomely, as it were) for the blades.

This same model applies to ink jet printers and cartridges. Ink jet printers are almost literally given away because the proprietary ink cartridges are very profitable. Protection of intellectual property (especially patents and trademarks) can apply to the printer and to the cartridges, including printing mechanisms and ink formulations. But the jewels are the patents that cover the way the cartridges mechanically and electronically mate to the base printer assembly (the interface).

You can also apply a patent to the way electrical contacts are made. Figure 14–1 depicts a photo of an electric motor on the left and a plug on the right.

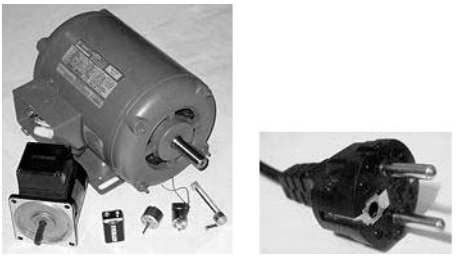

FIGURE 14–1 Deciding What to Patent

the right. Assuming both were new, on which device would you choose to get a patent? If you owned the plug, you could exclude (or tax) all possible innovations in motors. Or you could choose to keep the motor market for yourself.

There are many other examples of interface control. A microprocessor manufacturer that obtains a patent on some details of a chip socket or pin signaling can effectively exclude other chips from plugging into motherboards that support that socket. And this is true even if there is no infringement by the chip itself and the motherboard is freely available from others in the market. The encoding of data sent over the Internet (say, a video encoding scheme) and a protocol for accessing a service are other examples.

Clearly interface control is powerful. The limited monopoly afforded by a patent is amplified when an interface is involved. Essentially, the right-to-exclude concept, when applied to an interface, has the effect of *excluding all possible implementations*.

No doubt you can draw a few conclusions from this discussion. A purely business-motivated one could drive you to want to identify and control interfaces in your own engineering domain. After all, monopolistic profits are by far the richest ones. Another conclusion, and one we hope might just be tugging at your Citizen Engineer side, is that perhaps this is anti-innovative. Yes, if *you* happen to be the lucky (or smart) one to enjoy such powerful exclusive rights, you are likely to be pretty happy with the outcome. But if you want to enter the market with a new idea and are categorically excluded from doing so, you will think differently.

Most importantly, if you are a consumer or customer of the technology, you likely are paying too much and having your choices limited. That is, the societal interest in a system of controls that maximizes innovation fairly is not being well served. If you think of yourself as a Citizen Engineer this should bother you. We are using our skills to design things, and our societally

afforded system of IP controls acts in a way that might not be in the best interest of society.

Happily, there is something you can do. There is a basic set of concepts and principles that you can adopt that can align your interests with those of society. There are choices you can make in the way you approach the balance of creativity and control. We'll tell you how.

Innovation Commons

In his extraordinary book *The Future of Ideas,*[9] Lawrence Lessig identifies and explores the concept of an *innovation commons,* using the Internet as the exemplar of how intense innovation can flourish globally when the right balance between freedom and controls is struck. The book is also presciently cautionary about how accumulation of control points, often facilitated by our IP system, can throw a serious amount of sand in the gears—or worse.

Lessig describes how various layers can be composed, some that are proprietary (the physical infrastructure of the Internet is almost all privately owned) and other key layers that are free (e.g., the TCP/IP protocols for the communications layer and the HTTP and HTML standards for the Web layer). It's the mixture of these, and particularly the ability of the free layers to prevent monopolization of the private layers, that creates this globally accessible place for what is, by any measure, an enormous cycle of innovation.

There are also growing infrastructure and application layers in the Internet that are creating an even richer commons. The Linux operating system has been fundamental to the growth of the second generation of the Web. Also key have been free and open source application frameworks and services, notably the Apache suite of Web servers and the MySQL database.

The word *commons* is meant to evoke something akin to its figurative use: "The mutual good of all; the abstract concept of resources shared by more than one, for example air, water, information."[10] An innovation commons is an engineered place where we decide to collectively and freely contribute ideas and their expressions (e.g., specifications or code). By freely contributing, and then often collectively evolving, enabling interfaces and implementations, we get to do a number of things. Most important is that we serve society's interest by providing a fair place for innovation to thrive. That in turn serves our own business interests by providing even more opportunities to build and be compensated for implementations that are superior and/or add value on some important customer dimension.

Our use of the word *free* deserves attention. The freely available layers in an innovation commons are—to use Stallman's words again—"free as in freedom."

They might, and oftentimes are, "free as in free beer," but that is simply a technique to maximize adoption and participation by lowering barriers (financial ones). The freedom aspect is the key one. It's about the ability to *freely build upon each other's works,* the ability to share and innovate on top of, alongside, or in areas not originally anticipated. Again, and a central point here, is that the "free" layers aren't given away; they are contributed to the commons and are free in that context.

Open licenses play a defining role.[11] But they are just the beginning of the formation of a community. Eben Moglen captured the essence of this in his 2007 OSCON address, on the "Republic of Open Source." Here are Moglen's thoughts as transcribed by blogger John Eckman:[12]

> *Licenses are a part of community building—they are constitutions for communities. But the words of licenses are just the beginning—just as written constitutions are just the beginning of the republics which they give birth to and organize. In the 21st century it is no longer factories or individuals which are the units of production and distribution—it is communities. This is the reality of mass culture.*

We'll go into more detail in a moment on the secrets of keeping a community healthy. But properly ignited and maintained, the communities around an innovation commons have the potential to greatly expand the available market for all; a place of greater efficiency, faster cycle times, and competition. If we work together as Citizen Engineers to discover and build such free areas, we simultaneously serve market opportunities and society. We think that's a good thing.

The Economics of Open Source

We said earlier that mastery of intellectual property law is a great way to control the destiny of ideas—it's a powerful lever for getting people to participate in your ideas, and for using the ideas of others responsibly. But how does all that translate to better business results?

Richard Stallman was one of the first people to recognize that one great way to get people to participate with your stuff is to make your stuff free. "That's free as in freedom, not free beer," as he puts it. Basically, what Stallman was trying to do was to create open development communities: Others can take what you've done and they're free to build upon it and create new things, so you become part of something bigger. And so do they.

Great. But how do you stimulate that, and how do you make money at the same time? Let's look at the example of open source software. The thesis is that the open source model creates communities; communities create markets; then you can go in and monetize markets.

Red Hat is a great real-world example. This is a company built on a free software platform—a platform it didn't create. The software is available free of charge. Service and support will cost you—and Red Hat has built its business on support.

Or consider MySQL, the world's most popular open source database, used by virtually every Web 2.0 and e-commerce company today. The company had an extremely attractive *business* model (when we bought it in January 2008). MySQL had built a vibrant developer community that was millions strong and was known for a passion for disruptive innovation. Tapping into that community allowed Sun to enter new markets, drive new adoption of its hardware products, and open doors and create new opportunities for ISV partners and channel partners. It's a win for innovation and a win for business. They're not mutually exclusive.

Why is there a booming market for support of "free" software? Because at the enterprise level, every CTO wants to use free software, and no CIO will allow the use of free software without a commercial support contract. Not one. Downtime is just too expensive—millions of dollars per minute in some cases. So, they're smart to make sure they have someone standing behind the products they use to run their businesses.

On the other hand, if you can try a product before investing any money in it, why wouldn't you? The beauty of open source software is that customers can install it on their test servers free of charge, run it through any tests they like, and decide for themselves whether the program is all it's cracked up to be. They don't have to take a salesperson's word for it. Then, if they decide to deploy it, they can come back to you for a support contract.

There's no reason you can't offer a traditional commercial license as well. In fact, for competitive reasons, very few Internet services or embedded products companies will agree to reveal their code—a requirement with some of the open source licenses we discussed earlier. Such companies will gladly pay for a commercial license—and a support contract.

We believe open source is one of the most significant developments in network software collaboration. The world of network computing is bigger than any single company. It's a world that really requires the coordination and interaction and building of ecosystems that go across our entire industry and involve many, many people in many, many worlds. If you think about what we need in order to come together to coordinate, to make things happen, it's this community.

Beyond Software

Most of our reasoning about creativity and control, and the power of collaborative communities, is shaped by what has happened in software and, more broadly, the Internet. We certainly take liberties in extrapolating from open source software and inferring a more fundamental state for engineering and innovation: a freer state where innovation happens everywhere.

A fair question is whether such extrapolation is well founded. Could it be that there is *something special or different* about software development that lets that branch of engineering thrive under collaborative development, but not other branches? You might guess that we think what has happened in open source software is a leading indicator, a template as it were, for most every other branch in the collective art of engineering.

Why it started with software is easy to explain, though the reasoning is a bit circular. The Internet is the key *enabler* of global collaborative communities because it drives the cost of communication and sharing to near zero and because it can readily scale so that the essential network effects in these communities are allowed to take root and compound. We shouldn't be too surprised that the discipline at the foundation of the network itself, software engineering, is the first to take advantage of it. There also just might be a bit of good luck and fortune that the pioneers of the Internet came from an academic and research culture of extramural collaboration.

But again, perhaps there is something special about software and it simply doesn't generalize to other disciplines. We'll explore that concern here through a series of examples.

Goldcorp

In their book *Wikinomics: How Mass Collaboration Changes Everything,*[13] Don Tapscott and Anthony Williams chronicle the remarkable story of a struggling Canadian gold producer, Goldcorp, and its failed internal attempts to convert a wealth of geophysical data that suggested that it *should* be able to get more gold into actual production. The data suggested perhaps 30 times (!) the amount it was actually mining. But the precise interpretation of the data eluded corporate geologists throughout the 1990s.

In frustration, Goldcorp's CEO, Rob McEwan, decided on a radical path. He had just attended (in 1999) a conference at MIT on open source software, and wondered whether the same type of community participation model would work in the geophysics industry. His idea was to take all of Goldcorp's proprietary geophysical data accumulated since 1948—about half a gigabyte covering

55,000 acres of the company's Red Lake property—and make it freely available on the Internet. Moreover, he decided to fund what was called the "Goldcorp Challenge" with $575,000 in prize money, awarded to those who contribute to the discovery of new sources of gold on the property.

The community response was stunning. More than a thousand participants, ranging from practicing geologists to students to military officers, from more than 50 countries tried their hand at the data. A whole slew of new approaches and techniques were brought to bear, well beyond what had been attempted by Goldcorp's staff. In the end, 110 targets were identified, more than 50% of which were new. About 80% of the new targets yielded what ended up being 8 million ounces of gold. Goldcorp has subsequently become one of the most valuable companies in the industry today.

This style of bounty for community participation is being tried in a number of different settings. Notable is Google's "Android Challenge," which provided $10 million in awards for developers who built great applications for Android, the first open platform for mobile devices.*

TCHO Chocolates

TCHO is a San Francisco-based start-up that is attempting to revolutionize the way "obsessively good" dark chocolate is made using a range of open community strategies. At the consumer end, aficionados are invited to be "beta" testers of new recipes that are being developed in TCHO's laboratories. One of the key goals is to establish a flavor system, the TCHO Flavor Wheel, that lets consumers much more accurately reflect their taste preferences. The testing and evaluation are done via the Web in a highly participative manner. Both consumers and TCHO benefit from this; chocolate lovers have greater confidence that they will be buying something they will really enjoy, and of course, TCHO gets a much deeper understanding of what people will want to purchase.

Involving your customer via the Web is certainly not remarkable, but it is the next step in community building that makes TCHO a most interesting experiment in an innovation commons. TCHO is opening up the Flavor Wheel for participation by the cacao bean farmers themselves, many of whom have apparently never tasted chocolate nor know what the manufacturing process is like; as such, the farmers have been simply growing beans to get the highest yield against a fairly rudimentary system of quality grading. These beans

* Android Challenge is a contest sponsored by Google in which $10 million in prizes was awarded for mobile applications built on the Android platform (see http://code.google.com/android/adc.html).

get packed up and sold to wholesale chocolate producers, who in turn resell the bulk chocolate to "re-melters" who form most of the common brands. The current state of quality grading is likened to labeling a wine "France, 13% alcohol."[14]

Under the Flavor Wheel system, cacao farmers (for whom TCHO has provided Internet connectivity, incidentally) can see the market value of different kinds and qualities of cacao beans, and they can bias their production to where they can maximize their profits. TCHO is hoping the farmers form an open community of their own, where increasingly better methods are being tried and exchanged.

The Open Prosthetics Project

The concept of free and "open source" can apply to all forms of engineering, and mechanical design is no exception. Combine this with a core aspect of networked communities—the ability to bring together people under a common cause completely independent of whom and where you are—and you get some truly remarkable outcomes. One of these is the Open Prosthetics Project (OPP), which is part of an organization called the Shared Design Alliance. Their Web site explains:[15]

> *The Open Prosthetics Project is producing useful innovations in the field of prosthetics and freely sharing the designs. This project is an open source collaboration between users, designers and funders with the goal of making our creations available for anyone to use and build upon. Our hope is to use this and our complementary sites to create a core group of lead users and to speed up and amplify the impact of their innovations in the industry.*

A central focus of the OPP is the reengineering of a widely used, but no longer manufactured, arm prosthesis called the Trautman hook (which was introduced in 1925). Participants in the project contribute in all manner of ways (from designs, to testing, to ideas) in an effort to improve upon the original design. The "open source" aspect of this project is that these ideas can be freely shared. Jonathan Kuniholm, one of the founders and forces behind OPP, describes this purpose in an interview with *Scientific American*:[16]

> *"The reality . . . is that there's no traditional economic incentive to do work and make improvements on prosthetics. That doesn't mean that nobody cares, but most people don't have the money or know-how to magnify whatever efforts or improvements they make. I think we can generate*

> *far more societal benefit if we give away information than if we commercialized and sold the ideas. Our goal is to create a way to share these efforts and improvements with anyone who needs them."*

The project is clearly a successful example of the power of collaborative communities, though the connection with financially viable businesses is yet to be proven. In a real sense, the lack of IP protection makes it difficult for small companies to take the risk in manufacturing a design that anyone can freely reproduce. Nevertheless, there is a real social good, and a decidedly higher rate of innovation taking place than had in the past.

Wikipedia

Whatever you think about Wikipedia's reliability compared with "well-researched" sources, you should pay very close attention to the community model. In this regard, the success of the site is stunning. Wikipedia.org is well within the ten most popular Web sites, according to Alexa,[18] garnering around 250 million unique visitors each month.

This is the territory of significant commercial concerns, such as Yahoo!, Google, Microsoft, and Facebook. Yet fewer than a couple dozen people are employed by the Wikimedia Foundation, which operates on a shoestring budget. The secret, of course, is that there is an enormous amplifier in the Wikimedia contribution community, which the Wikimedia Foundation pegs at about 100,000 people, with a few percent being the most active. We think the licensing terms on the content have played a key role. As the Wikipedia entry on "Wikipedia" describes:[19]

> *All text in Wikipedia is covered by [the] GNU Free Documentation License (GFDL), a copyleft license permitting the redistribution, creation of derivative works, and commercial use of content while authors retain copyright of their work. The position that Wikipedia is merely a hosting service has been successfully used as a defense in court. Wikipedia has been working on the switch to Creative Commons licenses because the GFDL, initially designed for software manuals, is not suitable for online reference works and because the two licenses are currently incompatible.*

What is interesting to watch with this community is the way that collective content is continuously refined. All page entries are subject to "vandalism" whereby a rogue contributor provides deliberately misleading or biased

information. Interestingly, many of these contributions are corrected (by the community) in a matter of a few minutes, or perhaps a handful of hours.

We think that collaborative content development and maintenance is a key aspect of healthy communities, independent of discipline. The basic model popularized by Wikipedia looks to be a sound one and worthy of emulation.

OpenSPARC

If you peer over the shoulder of many digital hardware designers, you'd swear they were software engineers: They are in front of text editors writing code that looks like C. That code, written in high-level concurrent textual languages called Hardware Description Languages (HDLs),[20] describes the intended behavior of logical circuits. Many simulation and verification tools work on the high-level code, as do logic synthesis programs that essentially compile the code into digital hardware.

So, the concept of "open source hardware" is really a simple one: Develop the HDL code under an open source software license and try to foster open communities. That's exactly what we did at Sun with the OpenSPARC project, but there are many other examples as well—Opencores.org being a leading one.

With OpenSPARC, we released the HDL code (contained in thousands of files) under a GPLv2 license. We chose that license because we wanted to create a strong sense of sharing in the community. Not only does the license require put-back, but we are hoping that the viral aspect will encourage other "adjacent" contributions to be made as well (e.g., a new way of doing networking). The GPL also creates a commercial opportunity to sell the source code under license to those companies that, for competitive reasons, *don't* want to release any changes, improvements, or extensions they make to the design.

It's worth noting that, for high-performance digital systems, there is a big gap between the HDL code and a working chip. Certainly, automatic synthesis of entire chips is possible, but it's a lot of work involving a lot of engineering labor to get to an implementation that is competitive. So, there are natural barriers in this discipline to prevent someone from simply copying the physical design.

So far, we view the OpenSPARC program as enormously successful in building a global innovation community outside of Sun. There have been thousands of downloads of the design files and many universities have incorporated OpenSPARC into their curricula. At a minimum, we think this program illustrates how other engineering communities can be built around an open model.

Building an Open Source Community: Practical Advice from a Pro

"If you build it, they will come," whispers a voice from the cornfields in the movie *Field of Dreams*.

Many software engineers seem to hear the same voice. Having created some interesting code, they want to broaden its exposure. They want to entice others to use, amplify, propagate, and promote it. So, they create an open source community. And they learn the hard way that building a thriving community of users and creators can be more challenging and less likely to succeed than, say, building a professional-grade baseball field on a farm in Iowa.

We asked Kaj Arnö, who was instrumental in building the MySQL community (which we consider to be one of the most important communities in the annals of open source software), to offer some advice about building open source communities. Here's a summary of what he had to say.

Q: What makes a good candidate for an open source project?

A: You need to have a very cool technology; it needs to solve a core problem that many people share; and it needs to solve the problem better than anything else that's available.

Let's break that down, starting with the "cool" factor. A good test is, do people's jaws drop when they first see it or use it? Do their eyes open wide? If not, it may be simply an incremental improvement to something that's already out there, or it may not have broad appeal, or it may not be clear what the value is. In any of those cases, it's missing the spark that will ignite the interest in a community.

Now, it must also solve a common need in a novel way. You can bet that if you're working on a solution to a big problem, then someone else is too. Is your solution clearly better? If the advantage isn't obvious, it may be difficult to build a community around it.

Q: Assuming your software passes the "jaw dropping" test, why build a community? Why not continue to develop it yourself, or internally within your company?

A: A community can accelerate the propagation of your software dramatically. And I'm not talking only about using a community to recruit code contributors. Even in the MySQL community, not many people actually write code for us.

First of all, you don't need to explain all the basics to the people who join your community. In the case of MySQL, you don't need to tell them what a database is for. They know; they get it; they don't need time to come up to speed. They're ready to participate right now.

Second, the community is great for quality assurance. They want to help you test things, find bugs and list them, and so on. It's a very focused effort.

Third, as the community gains momentum it can deliver access to people who know a whole lot about the type of software you're working on. It becomes a highly concentrated source of a very specific expertise.

Q: What's the "right" way to go about building an open source community, and what mistakes do you see others make?

A: What you're trying to build is an architecture of participation. And you can't do that by blogging and giving away T-shirts. The key is not to "push" the community yourself or internally, but to keep the focus on building momentum externally.

Yes, it's a chicken-and-egg problem: You can't grow the community if you don't promote it, but the more actively you promote it the greater the risk of alienating people. A good starting point is to identify a set of core users who are highly informed and skilled, and spread the word. If your software has clear value, those people will quickly become evangelists.

At the same time, there are several things you can do to grease the wheels. First, do everything you can to make your software easy to get and easy to use. Make sure visitors to the community Web site can find the software, download it, and configure it easily. If they like what they're using, they'll see the value and find ways to contribute.

On a related note, make sure your documentation is good. If you can offer excellent documentation you will greatly increase the odds that people will want to work with your software. Documentation writers are a special breed of people—find good ones and keep them happy!

The design decisions you make relating to infrastructure are key. If you explicitly tie your software with a certain processor architecture or one vendor's systems, for example, you will limit the adoption rate of your software. My advice is to avoid tight integration with specific hardware whenever possible.

It's also important to have a good Web infrastructure for your project. As I mentioned, make it simple to find, download, and configure the software. But also make it easy to interact with others on the site. As the community grows, add forums, general usage discussions, and blog aggregations. Get a channel on Freenode. Start offering training and online support.

Q: But aren't some open source participants turned off by a glitzy Web site?

A: I think "slickness" can easily backfire because people associate it with a lack of substance. But there's nothing wrong with having an attractive Web site that's easy to navigate! There's a big difference between being efficiently functional and being glitzy. In the early stages of MySQL, there was absolutely no marketing on our Web site. Everything was technical; everything was written in a vendor-neutral way. I think people appreciated that, and we still try to focus on substance, objectivity, and honesty over style.

Q: What should you measure in assessing the community's growth?

A: Most organizations have the desire to measure and quantify the performance of everything, including community growth—but measuring can actually get in the way of growth if it's not done correctly or if you measure the wrong things. Always keep the focus on finding out how much users like what they're using. Two good measures are software downloads and documentation downloads. If they're using the software, if they're looking at the documentation, those are very good indicators that they like it. Other measurements, such as Yahoo! hits or click-throughs resulting from Google searches, can be misleading. You don't know if it's good word of mouth from other community members or clever search engine optimization techniques.

Other positive indicators could include bug activity, such as the number of filed bugs, verified bugs, and fixed bugs; the number of emails sent to the product email list—by both employees and nonemployees; blog and Wiki activity; and user popularity metrics such as the number of subscribers to your newsletter or email lists, downloads of the product on external sources such as SourceForge, [the] number of registered contributors, active users, signed contributor license agreements, and so on.

Q: What are some of the licensing issues you need to consider when building an open source community?

A: The choice of a license is just a step, because building a successful community of users and creators is the real prize. But the license decision is important because licenses are constitutions for communities. There's no general guidance I can offer—the choice will depend on the nature of your software and possibly your company's policies regarding IP ownership. Licenses such as BSD and MIT are very permissive; the GPL licenses are more restrictive and often require put-back and patent grants; and others such as Apache, CDDL, and Mozilla fall in between. Whatever license mechanism you choose, be clear about it and make sure your community knows exactly what it means for them.

Q: How can the founder of the open source community maintain control without being overly controlling?

A: Stay open and honest. In every community there will be participants who don't like the direction the project is taking, who have complaints about other participants, or who propose things that aren't in keeping with the community's values or are just bad ideas. Tell people what you think and you'll maintain control by maintaining respect. Also, don't feel compelled to be right all the time. Occasionally, I'll see a founder try to rule authoritatively, constantly asserting his or her superior knowledge. That approach usually leads to forking. If somebody's not happy, let them fork; usually if the fork is based on some personal motivation, it won't succeed. But in general, it's better to try to accommodate diverse viewpoints and encourage broader experimentation.