

open sources 2.0

Chris DiBona • Danese Cooper • Mark Stone



**THE CONTINUING
EVOLUTION**

O'REILLY®

Chris DiBona, Danese Cooper,
and Mark Stone

Introduction

Midnight comes to the Nevada desert, and nothing is visible but a line of taillights ahead and a line of headlights behind.

“You’ll see the lights from Gerlach first, and then Black Rock City,” the driver says to the passenger. They drive another 20 minutes in silence before the highway crests a ridge line. On the horizon, the blackness is broken by a band of multicolored light.

“Is that Gerlach?” the passenger asks.

“No.” The driver points to a small, dim cluster of yellow lights in the middle distance. “That’s Gerlach. Way out there, those lights are Black Rock City. We’re still about an hour away.”

The passenger ponders this for a moment, then asks, “An hour? How big is it?”

“For one week each year, Black Rock City is the fourth largest city in Nevada. Population 30,000, give or take.”

Gerlach rolls by, with its one bar, one gas station, and one motel. The caravan of cars bunches up after Gerlach. Then they turn off the highway, rumbling over the packed mud playa of the Black Rock Desert. Lights, neon, and thousands of RVs spread out before them. Music and drums—especially drums—can be heard in the distance. At the gate they’re approached by someone who looks like a transplant from Mardi Gras: face paint, bright-colored suit, and a carnival hat. He checks their tickets and flashes them a big grin.

“Welcome to Burning Man!”

* * *

When the original *Open Sources* was published in 1999, it served mainly as an affirmation that open source existed. The book brought together the leading voices in open source, demonstrating that we were a community, that we were indeed a movement to be taken seriously. To put that time in context:

- Microsoft had, only a year earlier, leaked the “Halloween Memo,” its first semi-public acknowledgment that open source was a competitive threat.
- IBM had provided some initial backing for Apache, but had yet to announce its \$1 billion Linux initiative.
- Linux was in only the 2.2 stage of kernel development.
- SourceForge.net was a relatively new site with only a few hundred projects hosted.

The mainstream press could not separate rising interest in open source from dot-com bubble hype. The media took the surface ideas of Eric Raymond’s *The Cathedral & the Bazaar* and created a caricature of legions of hobbyist programmers distributed across the globe, competing against the technology Goliaths of the day. That picture bore no more resemblance to reality than the tale of King Arthur does to the historical Middle Ages. Yet like any good mythology, it serves as a useful point of departure for understanding the real history from which it arose.

Today open source is an accepted fact of business life, with many companies engaged in core open source business models (Sleepycat, MySQL) or significant hybrid models blending open source and proprietary software (IBM, Novell, Red Hat). Many companies have striven to incorporate open source development into their range of software development practices—even Microsoft has projects hosted on SourceForge.net. How much do we really understand about the dynamics of open source software development or the communities that stand behind those projects?

The essays presented in this volume take a major step forward in our understanding since 1999, when the original *Open Sources* was published.

* * *

Burning Man is approaching its 20th anniversary. Conceived and inspired by Larry Harvey, it began in 1986 as a gathering of dozens of participants at Baker Beach in San Francisco. The centerpiece of the event was then, as it is now, the construction of a wooden effigy of a man, which is burned in celebration.

Celebration of what? The answer to that question may differ for every participant.

The event was originally timed with the summer solstice—it's now held the week leading up to Labor Day—and has always had a pagan, tribal feel to it. Participants began bringing their own art projects, many of which were also burned in celebration at the climax of the event. By 1990 the gathering numbered in the hundreds, and even the unusually tolerant San Francisco police made it clear that the event needed to find another venue.

Burning Man then moved to its current location—the Black Rock Desert—an empty stretch of Nevada desert on federal Bureau of Land Management land, roughly two hours north of Reno. The extreme remoteness and the harsh environment have become an indelible part of the event. To be there, you have to really want to be there.

* * *

Mitchell Baker makes clear in her essay that part of the strength of the Firefox community is its size. Thousands of people have contributed to Firefox, a community of contributors larger than the core project leaders can really envision. Firefox seems very much like one of those mythical “legion of programmers” projects that comes to mind when people think of the metaphor suggested by Eric Raymond's *The Cathedral & the Bazaar*.

Yet the open source development model remains the most enigmatic aspect of the open source community. One striking and unexpected outcome of the years since the original *Open Sources* is how little technology companies have been able to leverage the open source development process. Projects such as Linux and Apache have had world-changing success, yet no commercial software company has been able to replicate this development process for its own products or its own success. AOL, for example, has never figured out how to integrate the Mozilla/Firefox developer community into its product development process. Sun has struggled to open up both Java and Solaris, and the jury is very much out on the success of those projects.

Read the essays here by Chris DiBona and Jeremy Allison and you will see how little proprietary software development differs from open source software development. The differences their essays suggest are subtle: an emphasis on knowledge reuse, not just code reuse; a recognition that open standards matter; and that architecture needs to be created with openness in mind. Why, though, are these and other open source lessons so hard for commercial companies to use?

The real paradox is as old as Fred Brooks' classic, *The Mythical Man-Month*. In this work, Brooks formulates what has become known as Brooks' Law: that while the amount of programming work completed increases linearly as the number of programmers increases, the complexity of a project increases as the square of the number of programmers. The result is that large programming teams fail to reduce the time to project completion. The rationale is that the number of communication interfaces, which is roughly equivalent to the amount of coordination effort the project requires, increases geometrically as more people are added to the project.

Brooks' Law appears to set a fundamental limit on the optimal size of programming teams—and a rather small limit at that. Empirical evidence supports Brooks' Law. For example, since its inception SourceForge.net has maintained very close to a 10:1 ratio of registered users to registered projects, suggesting that open source development projects seldom have more than 10 active developers.

What are we to make, then, of the thousands of Firefox contributors? The key is to recognize that they are not a homogeneous mass of contributors, and Firefox is not a monolithic piece of software. In fact, the design is highly modular, enabling small teams to work on separate components of the code without interfering with each other. By far the most common characteristic open source projects have is a highly modular design. That design architecture is more a choice of social engineering than technical engineering, however. In the original *Open Sources*, Linus Torvalds commented that he started with a monolithic kernel design for Linux because he knew it would provide higher performance. Only later was Linux's distinctive system of loadable kernel modules developed, and it was developed as much out of a project management need as anything else. Could Apache provide higher performance as a purely monolithic piece of code? Probably. But the development process would become unmanageable. The original release of Mozilla suffered terribly because it was a monolithic piece of code. Firefox is the result of years of rearchitecting to achieve a modular, and thus manageable, design architecture.

Once it's clear that programming teams must necessarily be small, and that modularity is driven by communication and management needs more than by engineering needs, the structure of the open source community makes a lot more sense. The "bazaar" looks less like a bustling, homogenous mass and more like a structured community. More than anything, it resembles a tribe.

In fact, there is very little of Brooks' Law that is unique to software development. Any creative, collaborative knowledge enterprise faces the same constraints, and as a result, many collaborative communities adopt the same tribal structure.

* * *

Traditionally Black Rock City is laid out in a half-circle, with a grid of "streets," some of which form the spokes of the wheel, and others of which form concentric rings spreading from the center. At the very center is the Man, 40 feet of elegantly assembled wood, awaiting his night of conflagration at the end of the weeklong event.

A "burner" approaches a small tent encampment at the corner of 7 O'Clock and Justice. Out of the back of a pickup truck, two men from the encampment haul bags of sand and large rocks painted in black-light colors. They place the rocks inside a wooden sandbox and pour sand into it. Above them, a hand-painted sign reads "Reflections in Sand."

"So, what's your project?" asks the passerby.

One of the men looks up, squinting under the relentless desert sun. “We’re making a Zen garden,” he replies. He gestures toward a black light and generator lying nearby next to a couple of small, handmade wooden rakes. “Only this will be one you can enjoy at night.”

“Where are you guys from?”

“San Francisco.”

“You drove 350 miles to bring sand to the desert? Cool.”

Burning Man is a participatory event. According to its mission statement (http://www.burningman.com/whatisburningman/about_burningman/mission.html):

Our intention is to generate society that connects each individual to his or her creative powers, to participation in community, to the larger realm of civic life, and to the even greater world of nature that exists beyond society.

While not everyone who comes brings an art project, the expectation is that projects will be interactive in nature and that no one is there simply to observe. All are there to participate. Burning Man has no “audience.” While the event is not a political gathering, many projects have a definite message connecting to that “larger realm of civic life,” and to encourage this, each year Burning Man has an overall theme. Recent themes have included “The Vault of Heaven,” “Beyond Belief,” and “The Floating World.”

Several other ideas underpin the spirit of Burning Man. Black Rock City LLC has had a delicate relationship with the Bureau of Land Management over the years, striving to show that the event is a positive part of the area. A key part of this is Burning Man’s “leave no trace” philosophy. Art projects are not permanent, but are designed to be temporary, something can be removed or destroyed at the end of the event without leaving trash behind. Volunteers spend weeks after each event restoring the Black Rock Desert to its pre-event conditions. The “leave no trace” policy is not just a practical matter, though: it is also a good philosophical fit with the civic aspect of the event and with the fragile sense of the temporary that has been at the heart of the event since the first Burn.

Burning Man is also basically a nocturnal event. At roughly 6,000 feet elevation, the Black Rock Desert air is thin and dry. Daytime temperatures can hover around 100 degrees, so dehydration is the most common condition treated at the infirmary. As the sun goes down each day, people pause at whatever they are doing, look toward the craggy western hills, and cheer. Drums begin to beat. Music begins to play. Black Rock City comes alive.

Temporary, participatory, and nocturnal: those are the key elements of a Burning Man art project.

* * *

As puzzling as *how* open source projects organize themselves is *why*. To the casual outside observer, it appears that open source developers spend enormous amounts of time developing software that, in the end, they are simply going to give away without the prospect of compensation in return. While open source certainly has an altruistic side, altruism is neither the only nor the most important motivation.

First, one must realize that many open source projects started out of a developer's desire to solve an immediate problem. Linux was born of Linus Torvalds' desire to have a development platform on his PC at home. Apache came together from a group of people who had been relying on the NCSC web server and wanted to continue its development after NCSC stopped maintaining it.

Read Sonali Shah's essay, and you'll understand that the pattern here extends far beyond software development. Many consumer communities have come together collaboratively to innovate the products they consume, often when producers fail to produce innovation on their own. Software companies created just such a stagnant environment, out of which Linux and the rest of open source software was born. In this sense, the initial catalyst was quite selfish: developers wanted the software that companies were unwilling or unable to produce, so open source developers created it to "scratch their own itches."

What started from largely selfish motivations has evolved into something quite complex. In Steve Weber's essay, we get a clear analysis of just how complex the governance structures and processes of this community have become, as well as an intriguing view of where these enabling governance structures might foster collaborative communities in other endeavors. Andrew Hessel provides in his essay a very tangible example of where open source ideas are taking hold in an entirely new realm.

Lurking in the background, though, is still a question of motivation. If the inspiration for Linux was a selfish one, why make the choice to give the result away, and furthermore to do so under open source terms? How does selfishness become altruism?

The apparent paradox rests on the assumption that acts of charity necessarily conflict with acts of self-interest. From the point of view of a modern market economy, it often appears that charity and self-interest do conflict. What drives the open source developer, however, is clearly self-interest—even if it is based on an older notion of self-interest not easily captured by modern market economics.

The answer to the paradox lies in the reputation game played within the open source community. After all, monetary compensation is only a means to an end; it is a means of providing survival resources. Yet monetary compensation is not the only means to that end. While open source luminaries such as Linus Torvalds and Brian Behlendorf may not have the personal wealth of fortunate dot-commers, neither will ever lack for gainful employment. They have sufficient reputations based on their open source accomplishments to always be able to earn a living from their expertise.

Consider another fact: the largest age group among open source hackers is college students and graduate students: those under 25 for whom gainful employment is not an immediate issue, but one that certainly looms in their thoughts and plans. Because they are students, we can assume that they have the immediate survival resources needed for one to become a student in the first place. However, we can also assume that those survival resources are finite. Securing future survival resources is very much a part of the agenda of a student, indeed one of the main reasons for becoming a student in the first place. While a degree may provide a measure of that future security, a degree is not the exclusive means to that end: reputation as a creator of good code may provide that future security as well as or better than a degree.

Once this separation is made between monetary compensation and survival resources, we can see that there are historical precedents for this kind of behavior, and that there is a social model that loosely fits that of open source hacker culture. In Western civilization, we can look to medieval Europe, when nomadic groups like the Franks and the Vikings had settled into the agrarian-based feudal system. The mature feudal system of the 13th and 14th centuries has some interesting and instructive social structures; we'll focus on the concept of chivalry.

The good knight adhered to a code of behavior that transcended the laws of any particular kingdom and encouraged an attitude with some similarities to the attitude of today's hacker: a knight should be humble and should regard himself at the service of others, yet he would be judged by his prowess at his trade and would succeed to the extent that he could spread the reputation of his prowess.

Behind shield and visor, and upon adopting a particular set of heraldic emblems, a knight took on a kind of persona, creating a public identity that might be quite different from his private identity. There were regular events for testing and publicizing one's prowess at arms, such as the tournament or the hunt. There were orders of knighthood—again, often transcending the boundaries of kingdoms—that would certify one's prowess. To belong to such an order was an honor; to be of sufficient repute to be able to found an order and have other knights flock to it, was perhaps the greatest measure of success in the chivalric reputation game.

While a knight was part of the nobility, knighthood was a terrible burden financially. A suit of armor cost, relative to the medieval standard of living, the equivalent of a brand-new Mercedes today. Horses were expensive, and a knight was expected to have several. In addition, a knight had to maintain an entourage of squires, pages, and men-at-arms. He also officially owed 40 days of service to his feudal lord each campaign season, 40 days that in reality could often drag into several months. A landed knight with a small manor could easily spend any excess capital just to maintain his position; a landless knight, or knight errant, would likely live in perpetual debt.

What motivation, then, would a young man in the Middle Ages have to aspire to knighthood? Did one live according to the code of chivalry out of pure selflessness, and a desire to serve others? Or was there some more pragmatic force at work? To a knight of repute, money was not really important. His lord, or anyone else interested in retaining his services, would see that his needs were met, that he had survival resources. To flourish, a skilled squire aspiring to knighthood need only hone his skills and act to establish and further his reputation.

This attitude and this kind of behavior seem quite similar to that of the young student who is an aspiring hacker. While academia, and academic computer science, is a reputation game of its own, what is fascinating about computer science is that there is a large body of practitioners that refuse to play this particular reputation game. To this latter group, an education really is just a means to an end, and the end is to develop the skills necessary to create good code. Some may go far enough to pick up a university degree, but many do not, and the degree is clearly secondary to the ability to code. In other words, one can build a reputation without having to acquire the academic pedigree. It is not simply a distaste for academia that fosters this kind of behavior. The hacker who is more interested in picking up skills than in picking up a degree is often the same hacker who is unwilling to be tied down to a steady job, preferring to move from assignment to assignment as a freelancer and consultant. These are the knights errant of the open source movement.

In reality, the medieval knight errant was essentially a mercenary, hardly the noble figure portrayed by Malory or Chaucer. The fact that these soldiers were mercenaries made chivalry no less important. A mercenary captain had to be trustworthy; his word of honor alone was a binding contract. Otherwise, he simply was not employable. These men lived by the chivalric code. What they found was that that code alone assured them survival resources. A skilled and honorable mercenary captain was never without employment and never lacked for resources.

Chivalry and pragmatism are not conflicting goals, but that pragmatism can indeed be served by chivalry. The mercenary captain lacked money, land, and all other tangible resources. He had only one form of collateral: his reputation. That reputation could be maintained only if his behavior was seen to be genuinely honorable. Chivalry, then, was a necessity: it was an essential ingredient in building the only available collateral that could be parlayed into survival resources.

From this analogy, we can learn several lessons about today's hackers. First, the open source gift culture need not be seen as strictly, or even predominantly, altruistic. Pragmatism and altruism are not mutually exclusive. Today's hackers, like the knights errant and mercenaries of old, can and do trade in the coin of reputation as a means of achieving survival resources.

Second, as a culture matures, the pragmatism becomes more apparent. This was true in the Middle Ages. In the high Middle Ages, the era of the crusades, the knight errant made a flamboyant pretext of making a gift of his skills and services; one has to look below the surface to see a rational exchange of skills for resources in such gift acts. By the late Middle Ages, though honor and reputation were still essential, when a nobleman retained the services of a mercenary captain, the transaction was explicitly and without apology for the mutual benefit of nobleman and captain.

We see this same trend among today's hackers. While reputation alone has always provided survival resources for some, the trend to switch the meme about their activity from "Free Software" to "Open Source" reflected a maturing shift from altruistic pretext to honest pragmatism.

Today we see a symbiotic balance between the chivalric open source hackers and the companies that employ them. In fact, this development is foreshadowed in the events Eugene Kim describes in his essay, which concerns the development of the first commercial compilers. Even in the 1950s it was possible for a company such as IBM to see the advantages of transforming a competitive relationship into a collaborative one.

Today a number of prominent open source developers are employed at major technology teams. Novell transformed itself in a matter of months into a major player in the enterprise open source space through the acquisition of Ximian and SuSE. Of contributors to this volume, Jeremy Allison works for HP, and Chris DiBona works for Google. Neither works at an open source company per se, but both have an understanding that it is in their employer's interest that they be allowed time and resources to continue working on open source projects. Other contributors here, such as Ian Murdock (the "ian" of "debian"), Michael Olson, and Stephen Walli, are involved in more purely open source business models.

* * *

Burning Man is, in some sense, a commercial operation. There is a significant admission charge for the event (more than \$200), and the event is run by a limited liability corporation. The corporation's main purpose, however, is to sustainably manage the event. There is a permit to obtain from the Bureau of Land Management every year. There is insurance for the event. There is preparation before and cleanup after the event, as well as basic infrastructure, such as sanitation services, that must be provided every year. Finally, there is a small paid staff responsible for everything from event promotion and organization to informal lobbying efforts with the Department of the Interior, Washoe County, and the state of Nevada.

Once inside the gates, however, participants are forbidden from engaging in monetary commerce. The primary form of commerce is barter. In the spirit of the event, barter is as much a pretext for participation as an exchange of goods. It may take the form of a scavenger hunt, where admission to an art project requires a ticket stamped

by several other art projects. It may take the form of a raid by the “Viking Longship” art car, which “pillages” camps but always leaves some small gift behind. Or it may be in the form of a quiet bar on a back street of Black Rock City that asks only some small trinket from the day’s events as the price of a drink.

Unfettered from monetary exchange, however, most denizens of Burning Man gravitate toward a gift economy. Acts of giving range from the mundane to the extravagant: the accordion player who serenades those in the porta-potty line with his renditions of AC/DC; the massage therapist volunteering her services; the water-gun brigade, spraying people down for a moment of cool relief from the midday sun; or the man who brings along a week’s supply of dry ice so he can serve cold ice cream every day.

* * *

One of the most ironic developments since the publication of the original *Open Sources* has been the rapid adoption of open source business models by technology companies.

In 1999, the consensus view of the business community was that giving away intellectual property for free was a poor basis for doing business. At that point, Michael Olson and Sleepycat Software had been quietly pursuing their dual licensing model for Berkeley DB for three years. Now several open source database companies are pursuing similar models.

Sleepycat’s approach is an example of the more general business dynamics at work behind open source. One of the key effects is commoditization, discussed in different aspects in essays by Matt Asay, Ian Murdock, and Stephen Walli. Commoditizing a complement to one’s core business serves to enhance that business. It brings down the cost of entry for customers, thus expanding the potential market size for the core business. The key is to have a complementary core that can be monetized. Sleepycat achieves this through dual licensing, charging for a proprietary license for customers who are unwilling or unable to open their own source. Novell achieves this through a hybrid business model, with a service business and a proprietary software business further up the “application stack” from its commoditized Linux business.

Commoditization is not the only benefit. Open source business models lower the cost of both sales and marketing. The common fear with any free product is that “you get what you pay for.” With open source, however, the source code is entirely open to inspection so that there are no hidden surprises. Further, the source code can be freely redistributed. Several market effects result from this. First, those most likely to avail themselves of open source are those with the greatest understanding of its benefits—namely other software developers who actually have the skill and desire to examine source code. Second, the distribution model creates a user community of like-minded enthusiasts without intervention or incurred marketing costs by the

originating company. Finally, that user base will, at some point, approach the originating company with a request for additional features, services, or complementary software. In other words, by its very nature, open source has very low marketing costs that create an inbound sales channel of prequalified leads.

Open source software companies that exploit this dynamic can thus maintain lower overall operating costs, consequently passing on lower prices while still maintaining healthy profit margins. All of this accelerates the commoditization process, making a well-established open source software product quite difficult to compete against.

Yet these very business advantages inherent in open source bring us to another aspect of the same paradox: why is it so difficult for companies to leverage open source as a development model, rather than as a business or marketing model? Consider Sleepycat's dual licensing scheme. The model works only if Sleepycat holds full copyright to all of the software in Berkeley DB. Otherwise, it is not permitted to offer the second, proprietary license in addition to the open source license. If it must have all rights to the software, though, that means that the software must essentially be developed in-house. Sleepycat does its own development instead of leveraging outside, open source development.

Perhaps Russ Nelson offers a purer example of an open source business model, one where he both develops open source software and leverages the open source developments of others. The complimentary values Russ Nelson offers are his reputation and his expertise, both carefully maintained over the years. The resulting business may not be a large one, but it is one where he alone is the master of his own destiny.

* * *

Burning Man certainly has the feel of an organic, grass-roots movement. Certainly that grass-roots element is part of the dynamic that makes the Burning Man community what it is. But simply thinking of "grass roots" makes it too easy to overlook what a complex community structure Burning Man has and requires.

First, there is the structure of Black Rock City itself. Maintaining order in Black Rock City is primarily the responsibility of the Black Rock Rangers. They describe themselves as a "non-confrontational mediating agency" (see <http://www.rangers.org>). They are all volunteers, and they have no official authority. They pay admission like everyone else; rangers is not a way around the admission price. Further, each ranger is required to attend training prior to the event, and each ranger must enter the ranger program with the sponsorship of another ranger as mentor. While the rangers occasionally call in actual law enforcement, for the most part the rangers are accorded tremendous respect.

Black Rock City has many of the elements of any other city of 30,000. There is a radio station, some years several. There is a DMV (Department of Mutant Vehicles); you cannot drive around within Black Rock City or the playa beyond without registering with the DMV. There is an airport; dozens of attendees routinely fly in for

Burning Man. And of course, there is a newspaper, the *Black Rock Gazette*, published six times during each Burning Man event.

Residential areas of Black Rock City have structure as well. Look at a map of Black Rock City when you arrive at Burning Man, and you'll see a number of areas along the inner circle marked as Theme Camps or Villages. These are areas that are both residential and interactive, involving a large number of people working on a common art project where the residential area itself is the art project.

The inner circle faces toward the Man, and beyond on the playa is the Burning Man "gallery" of art installations. Here is where the larger projects are constructed and the larger group events are played out. Typically, there will be an opera and several other stage performances. Weddings are common.

For several years, there was a project called Solaria. It was a scale model of the solar system, where not only the distances between objects were proportional, but also the size of those objects relative to distance was proportional. Each object was a light source, with the sun represented by a small lamp about the size of a bowling ball. On that scale, Pluto could be reached only by a three-mile bike ride across the playa. Not even the Smithsonian can put on an exhibit of that scale.

* * *

No one has grasped the power of commoditization as quickly as developing nations. This is an international arena with general concern over globalization, anxiety about domination by American corporations, and fear of Microsoft's monopoly in particular. Open source has given developing nations a bargaining chip to pressure technology companies, especially Microsoft, on price. The natural response has indeed been a lowering of prices in countries ranging from Brazil to India.

Yet the significance of open source goes far beyond commoditization and price pressure. Read the essays here from Jesus M. Gonzalez-Barahona and Gregorio Robles, Alolita Sharma and Robert Adkins, and Boon-Lock Yeo, Louisa Liu, and Sunil Saxena, and you see that open source is really about controlling one's technology destiny. Outside the United States, people find it odd that we use the same word, *free*, to mean two very different things: *with no cost* or *liberated*. The open source community has adopted the slogans "free as in beer" versus "free as in speech" to draw attention to the difference. Commoditization is all about "free as in beer;" what developing nations care about the most is "free as in speech."

Open source provides greater intellectual property control than proprietary software that one does not own. Developing nations want control over the intellectual property on which their technology infrastructure depends. What emerges is a different sense of ownership from the traditional market economy sense of ownership, one that speaks not just to the motivations of policymakers in developing nations, but to the motivations of open source developers as well. Think again of chivalry and of our feudal heritage.

In a feudal system, a farmer could not own land, nor the harvest from that land. Serfs were indentured to the land and were entitled to only a portion of their harvest after paying their taxes to the feudal overlord and landowner.

Technology workers today face an analogous form of servitude. It is almost universal practice at technology companies to confront new employees with a hiring agreement that says, among other things, that any and all code and inventions created by the employee while in the employ of the company belong to the company; all copyrights and patents resulting from these creations must be transferred to the company. Technology workers may reap the fruits of their creative labor only under terms dictated by the company. Our modern notions of intellectual property and ownership of it are based on this relationship: that it is fundamentally companies, not individuals, that own intellectual property, and that individuals create new intellectual property primarily in the service of companies.

If open source hackers have one common attitude that ties them together into a community, it is the rejection of this notion of intellectual property. The conventional outsiders' view is to say that open source software is not owned. It is fear of the lack of accountability associated with this perceived lack of ownership that makes many companies reluctant to deploy open source software for mission-critical functions.

In fact, this conventional view is deeply mistaken. To understand why, we must make a distinction between “ownership” and “stewardship.” Ownership is something that is fully transferable from one owner to another without loss of value. Money, and many (though not all) material objects, are examples of entities that can be simply owned. Stewardship, on the other hand, applies when something undergoes change, when it evolves, or when it has some kind of life cycle. In this case, something has value only if it is cared for in such a way as to sustain the life cycle. In an agrarian society, animals are a prime example of something requiring stewardship. Skills are required, and effort must be put forth, to maintain a herd. Transferring the herd to someone who lacks those skills or is unable to put forth the effort diminishes the value of the herd. In other words, only a good steward can realize the full value of that which is stewarded.

Historically, land has been, and continues to be, at the center of contention between these two notions of ownership. For example, Native Americans considered themselves stewards of the land, and thus fell victim to the European notion of landownership. Today the battle between environmentalists and certain corporations is over exactly these two conflicting senses of ownership. Andrew Hessel's essay points to a brewing conflict in these senses of ownership with the biotech industry.

In the technology sector, open source developers believe that software requires stewardship. The standard employment contract and the proprietary software it engenders preclude stewardship. Open source software, however, by its very nature

encourages stewardship. Again, the motivation here is not altruism or charity. To an open source developer, stewarding software is the best way to see that the software evolves and improves, and hence it's just pragmatism to take a stance toward intellectual property that assures that software will be stewarded.

The proof is in the longevity of open source software projects and the stewards who tend them. Linus Torvalds is still at the head of the Linux kernel “tribe” more than a decade after the first public release of Linux. Eric Allman has guided Sendmail for more than 20 years. Larry Wall is still the guiding vision behind Perl, again after more than 20 years. In these and many more cases, a common core group stood behind the software for far longer than most proprietary software enjoys the benefits of a common development team. It is this—the dynamics of stewardship—far more than the “legions of programmers” that accounts for the success of open source software.

Further, it is this dynamic of stewardship that fosters the social network around open source software that is based on the reputation game. Having committed themselves to what they regard as the most pragmatic approach to intellectual property, open source hackers have then adopted the professional and social structure needed to support that approach to intellectual property. It isn't altruism. It's chivalry, a far subtler and more pragmatic thing.

* * *

The Burning Man event lasts for one week each year. Contained within that event is the remarkable community of Black Rock City. Yet it would be a mistake to assume that the community exists for only one week a year. The complex structure and intricate hierarchy that is the Burning Man community could not adapt, evolve, and sustain itself if it were not a year-round phenomenon.

That is the deeper truth not obvious to the casual observer: Burning Man is a permanent worldwide community whose members are connected to and engaged with each other continuously. Art projects that are on exhibit at Burning Man will often be shown at smaller gatherings in places such as San Francisco. Burners gather for regular social events throughout the year to talk about past events and plan for next year. When Washoe County or the Bureau of Land Management considers a change that may affect the permit for Burning Man, word travels through the community like wildfire, and burners show up in force to make their cases and state their views.

It's no accident that the growth of Burning Man parallels the growth of the Internet. The Burning Man web site is an impressive knowledge archive about the event and the community, providing a wealth of information and resources for anyone trying to understand Burning Man and learn how to get involved. The “Jack Rabbit Speaks” mailing list provides an announcement forum that goes out to the whole of the Burning Man community, but there are dozens of other mailing lists that tie together smaller communities within. Some of these are organized by geographical proximity,

but many more are organized by common interest. A theme camp will often have a mailing list for its members. The Black Rock Rangers have their own web site, and their own mailing lists.

What the Internet has done is to free us from the constraints of geography in terms of whom we connect to, whom we share common interests with, and whom we form community with. The power of intentional community is abundantly clear in the open source movement, where developers from around the world can collaborate on software of common interest. Yet the larger lesson is that the power of collaboration and the power of community exhibited in open source have relatively little to do with software development.

We begin to see the lasting significance of open source only when we see that it is one instance of a general pattern of online, collaborative community. Even a very physical, tangible event like Burning Man is crucially dependent on this larger sense of community. If we look closely, we see that this pattern of collaboration is beginning to manifest itself in many other places beyond open source.

We see this strikingly in Eugene Kim's essay, contrasting an early example of software collaboration with the grass-roots collaboration that emerged around the Ground Zero cleanup. We see this in the power of consumer-driven innovation that Sonali Shah explores. We see it in the sense of community behind Slashdot, described by Jeff Bates and Mark Stone, and the spontaneous movement that became Groklaw, described by Pamela Jones. And we see a compelling attempt to distill the most general patterns of these communities in Steve Weber's essay.

The simplest elements are these:

- Recognizing that one has common cause with people who might otherwise have competing or divergent interests
- Acknowledging that small teams working on a component of a problem are the only scalable way to tackle large problems
- Improving solutions iteratively through a sense of stewardship, ecosystem, and evolution, rather than a sense of property and ownership

Taken together, these principles suggest an organizational structure that is at once novel and familiar. These intentional communities form hierarchically, but it is a hierarchy based on achievement and reputation rather than power, money, or authority. Communication flows easily up and down the hierarchy, but decision-making flows from the top down. "Everyone gets a voice, but not everyone gets a vote" (see Bates and Stone). The resulting organization is more tribal than democratic.

* * *

Each night when the two burners return to camp in the hours before dawn, “Reflections in Sand” has changed shape. Some new pattern has been carefully raked into the black-lit sand, and though they never see the visitors, the evidence of their passage and participation is there.

The last evening is the night of the Burn. All day the Man has been laid down flat as he is prepped. At sunset he is raised up again. While all week his arms have been down at his side, now they are raised high above his head. This is the signal for the ceremony to begin.

A ring of lights surrounds the Man, and the rangers walk the perimeter to ensure everyone keeps their distance. For hours, within the ring of lights, drummers, firedancers, and musicians perform. Pagan rituals from 1,000 years ago must not have looked so different. When at last the Man ignites, flames shooting 50 feet or more into the night sky, there is awed silence. Pieces begin to fall off, and he begins to tremble, as only guy wires hold him in place. The trembling increases, and at last the whole Man collapses into a burning mound. At that moment the crowd rushes the center in a wild, swirling dance that brings them as close to the flames as heat will permit.

Now when the two burners return to their camp, they find the sand has been raked again, into one last new pattern. And something more; there, in the very center of the little Zen garden, someone has left a bottle of water. One of them reaches in to retrieve the bottle and pulls off the lid.

“You going to drink that?” asks the other.

“Of course. Out here, water is the most precious gift of all.”

An hour later their camp is packed. The black-light-painted rocks go back with them, the sand has been scattered, and the wooden frame and rakes have been heaped onto their neighborhood burn pile. They drive toward the gate, and the attendant waves them down.

“Heading out?” asks the attendant.

“Yeah, back to San Francisco,” replies the driver.

“This your first time here?”

The passenger answers, “It is for me.”

The attendant gives a knowing smile and waves them through. “See you next year. Welcome to the tribe!”