Tim O’Reilly
in a Nutshell
## Contents

### Introduction

### Part I. On Technology

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The New Era of Pervasive Computing</td>
<td>12</td>
</tr>
<tr>
<td>All Software Should Be Network Aware</td>
<td>14</td>
</tr>
<tr>
<td>The Open Source Paradigm Shift</td>
<td>17</td>
</tr>
<tr>
<td>Watching the “Alpha Geeks”: OS X and the Next Big Thing</td>
<td>46</td>
</tr>
<tr>
<td>Hardware, Software, and Infoware</td>
<td>55</td>
</tr>
</tbody>
</table>

### Part II. On Publishing

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piracy is Progressive Taxation and Six Other Lessons on the Evolution of Online Distribution</td>
<td>68</td>
</tr>
<tr>
<td>Information Wants to be Valuable</td>
<td>81</td>
</tr>
<tr>
<td>Beyond the Book</td>
<td>86</td>
</tr>
<tr>
<td>Buy Where You Shop</td>
<td>93</td>
</tr>
</tbody>
</table>
Part III. On Business and Life

Rules of Thumb 98
Fishing with Strawberries 101
Some Necessary Qualities 103
Ten Years as a Publisher 106
Knowing When to Let Go 110
Walking the Kerry Way 113
Books That Have Shaped My Thinking 119
Introduction

A preacher. That’s what the high school aptitude test told Tim O’Reilly he should be. Possessed of a yearning to understand, a passion for doing the right thing, the desire to spread the good word, and the skill to persuade, Tim was a prime candidate for the ministry. As it happens, he ended up preaching the gospel of technology.

Tim is also, fundamentally, a writer. In the twenty-five years he’s spent immersed in the computer technology world, Tim has often turned to writing as a tool to understand, and to share what he discovers. *Tim O’Reilly in a Nutshell* is a collection of Tim’s most thought-provoking pieces about technology, the publishing business, and life (arranged by theme rather than date). For more of Tim’s writing, see *tim.o reilly.com*. 
On Technology
This could have been easily dismissed as reckless optimism when I published it on tim.oreilly.com. Just over a year later, the signs of renewed, sustainable technology growth are starting to manifest. The future’s never certain, but my optimism is firmly intact.

We’re now ten years into a massive technology-driven transformation of the social, political, and economic landscape. After the dot-com bubble and subsequent meltdown, we think we’ve seen it all. But in fact, as Carlota Perez tells us in her classic book, *Technological Revolutions and Financial Capital*, every major technological revolution for the past three hundred years has been accompanied by a financial bubble. It is typically only after the bubble is over that the long-term impacts of the technological change begin to be felt. A radical new technology is deployed in a wave of irrational exuberance. After the inevitable bust, the infrastructure that was created during the bubble becomes the basis for a long period of steady growth as the technology is really put to use. Short-term thinkers flee the market just when things are getting interesting, leaving the gains to be reaped by those with their eye on the long view.

So with that long view in mind, here are some of the reasons why I’m still bullish on our industry.
Java™ and open source technologies like Linux®, Apache, MySQL, and Perl/PHP/Python (LAMP) are just hitting their stride and will become even more important over the next decade. Years ago, I urged Linux developers not to focus on displacing Microsoft, but on becoming the “Intel Inside” of the next generation of computer applications. And we’re well along the path to that vision. Along with Java-based platforms like BEA WebLogic® and IBM’s WebSphere®, LAMP-based applications are at the heart of today’s “killer applications”—Google, Amazon, Yahoo!, and a host of other familiar web titans.

On the enterprise side, a web services infrastructure is tying businesses together with applications built on what is gradually shaping up to be a next-generation “Internet operating system.”

Meanwhile, Microsoft is not to be counted out. Despite their recent backpedaling on .NET marketing, Microsoft’s eyes are clearly focused on the future, and a tactical retreat on overheated marketing does not mean that they are quitting the game—only that they are about to try rewriting the rules once again.

At the same time, Mac OS® X, Steve Jobs’ vision of the digital hub, and new handheld devices are all radically changing consumer expectations.

The era of the personal computer is over. We are entering the pervasive computing era, where dozens or hundreds of specialized access devices suck services from the emergent global network computer. At O’Reilly, we’re working hard to prepare you for that future, as you take the tools we teach you about and use them to invent it, in a virtuous circle.
All Software Should Be Network Aware

October 2003

Sun’s prescient marketing slogan, “The Network is the Computer,” is becoming a fact of life. In this piece, written for our Fall 2003 catalog, I posit that we need to re-think what software is, where it lives, and how it works for users.

I’ve been noodling for some time on the idea that we need some kind of equivalent to the old Apple Human Interface Guidelines for the new world of networked applications. The original Human Interface Guidelines laid out Apple’s vision for a set of consistent approaches for GUI applications. Even though Windows® ended up with a different set than the Mac®, the idea was simple and profound: create a consistent set of user expectations for all applications and live up to them. Now that we’re moving into the era of “software above the level of a single device” (to borrow Dave Stutz’s phrase), we need something similar for network-aware applications, whether those applications live on a PC, a server farm, a cell phone or PDA, or somewhere in between.

Here are some of the things that I’d like to see universally supported:

1. Rendezvous-like functionality for automatic discovery of and potential synchronization with other instances of the application on other computers. Apple is showing
the power of this idea with iChat® and iTunes®, but it could be applied in so many other places. For example, if every PIM supported this functionality, we could have the equivalent of “phonester” where you could automatically ask peers for contact information. Of course, that leads to guideline 2.

2. If you assume ad-hoc networking, you have to automatically define levels of access. I’ve always thought that the old Unix UGO (User, Group, Other) three-level permission system was simple and elegant, and if you replace the somewhat arbitrary “group” with “on my buddy list,” you get something quite powerful. Which leads me to . . .

3. Buddy lists ought to be supported as a standard feature of all apps, and in a consistent way. What’s more, our address books really ought to make it easy to indicate who is in a “buddy list” and support numerous overlapping lists for different purposes.

4. Every application ought to expose some version of its data as an XML web services feed via some well-defined and standard access mechanism. One of the really big wins that fueled the early Web was a simple naming scheme: you could go to a site called www.foo.com, and you’d find a web server there. This made web addresses eminently guessable. We missed the opportunity for xml.foo.com to mean “this is where you get the data feed,” but it’s probably still possible to come up with a simple, consistent naming scheme. And of course, if we can do it for web sites, we also need to think about how to do it for local applications, since . . .

5. We ought to be able to have the expectation that all applications, whether local or remote, will be set up for two-way interactions. That is, they can be either a source or sink of online data. So, for example, the natural com-
plement to Amazon’s web services data feeds is data input (for example, the ability to comment on a book on your local blog, and syndicate the review via RSS to Amazon’s detail page for the book). And that leads to:

6. If data is coming from multiple sources, we really need to understand who owns what, and come up with mechanisms that protect the legitimate rights of individuals and businesses to their own data, while creating the “liquidity” and free movement of data that will fuel the next great revolution in computer functionality.

7. We need easy gateways between different application domains. I was recently in Finland at a Nokia retreat, and we used camera-enabled cell phones to create a mobile photoblog. That was great. But even more exciting was the ease with which I could send a photo from the phone not just to another phone but also to an email address. This is the functionality that enabled the blog gateway, but it also made it trivial to send photos home to my family and friends. Similarly, I often blog things that I hear on mailing lists, and read many web sites via screen-scraping to email. It would be nice to have cross-application gateways be a routine part of software, rather than hacked on after the fact.

As networked devices become ever more central to our lives and work, I hope to see even more “network aware” functionality routinely incorporated into new applications and devices.
This article is based on a talk that I first gave at Warburg-Pincus’ annual technology conference in May of 2003. Since then, I have delivered versions of the talk more than twenty times, at locations ranging from the O'Reilly Open Source Convention, the UK Unix User's Group, Microsoft Research in the UK, IBM Hursley, British Telecom, Red Hat's internal “all-hands” meeting, and BEA's eWorld conference. I finally wrote it down as an article for an upcoming book on open source, Perspectives on Free and Open Source Software, edited by J. Feller, B. Fitzgerald, S. Hissam, and K. R. Lakhani; to be published by MIT Press in 2005.

In 1962, Thomas Kuhn published a groundbreaking book entitled The Structure of Scientific Revolutions. In it, he argued that the progress of science is not gradual but (much as we now think of biological evolution) a kind of punctuated equilibrium, with moments of epochal change. When Copernicus explained the movements of the planets by postulating that they moved around the sun rather than the earth, or when Darwin introduced his ideas about the origin of species, they were doing more than just building on past discoveries, or explaining new experimental data. A truly profound scientific breakthrough, Kuhn notes, “is seldom or never just an increment to what is already known. Its assimilation requires the
reconstruction of prior theory and the re-evaluation of prior fact, an intrinsically revolutionary process that is seldom completed by a single man and never overnight.”[1]

Kuhn referred to these revolutionary processes in science as “paradigm shifts,” a term that has now entered the language to describe any profound change in our frame of reference.

Paradigm shifts occur from time to time in business as well as in science. And as with scientific revolutions, they are often hard fought, and the ideas underlying them not widely accepted until long after they were first introduced. What’s more, they often have implications that go far beyond the insights of their creators.

One such paradigm shift occurred with the introduction of the standardized architecture of the IBM personal computer in 1981. In a huge departure from previous industry practice, IBM chose to build its computer from off-the-shelf components, and to open up its design for cloning by other manufacturers. As a result, the IBM personal computer architecture became the standard, over time displacing not only other personal computer designs, but over the next two decades, minicomputers and mainframes.

However, the executives at IBM failed to understand the full consequences of their decision. At the time, IBM’s market share in computers far exceeded Microsoft’s dominance of the desktop operating system market today. Software was a small part of the computer industry, a necessary part of an integrated computer, often bundled rather than sold separately. What independent software companies did exist were clearly satellite to their chosen hardware platform. So when it came time to provide an operating system for the new machine, IBM decided to license it from a small company called Microsoft, giving away the right to resell the software to the small part of the market that IBM did not control. As cloned personal computers were built by thousands of manufacturers large and
small, IBM lost its leadership in the new market. Software became the new sun that the industry revolved around; Microsoft, not IBM, became the most important company in the computer industry.

But that’s not the only lesson from this story. In the initial competition for leadership of the personal computer market, companies vied to “enhance” the personal computer standard, adding support for new peripherals, faster buses, and other proprietary technical innovations. Their executives, trained in the previous, hardware-dominated computer industry, acted on the lessons of the old paradigm.

The most intransigent, such as Digital’s Ken Olson, derided the PC as a toy, and refused to enter the market until too late. But even pioneers like Compaq, whose initial success was driven by the introduction of “luggable” computers, the ancestor of today’s laptop, were ultimately misled by old lessons that no longer applied in the new paradigm. It took an outsider, Michael Dell, who began his company selling mail order PCs from a college dorm room, to realize that a standardized PC was a commodity, and that marketplace advantage came not from building a better PC, but from building one that was good enough, lowering the cost of production by embracing standards, and seeking advantage in areas such as marketing, distribution, and logistics. In the end, it was Dell, not IBM or Compaq, who became the largest PC hardware vendor.

Meanwhile, Intel, another company that made a bold bet on the new commodity platform, abandoned its memory chip business as indefensible and made a commitment to be the more complex brains of the new design. The fact that most of the PCs built today bear an “Intel Inside” logo reminds us of the fact that even within a commodity architecture, there are opportunities for proprietary advantage.

What does all this have to do with open source software, you might ask?
My premise is that free and open source developers are in much the same position today that IBM was in 1981 when it changed the rules of the computer industry, but failed to understand the consequences of the change, allowing others to reap the benefits. Most existing proprietary software vendors are no better off, playing by the old rules while the new rules are reshaping the industry around them.

I have a simple test that I use in my talks to see if my audience of computer industry professionals is thinking with the old paradigm or the new. “How many of you use Linux®?” I ask. Depending on the venue, 20-80% of the audience might raise its hands. “How many of you use Google?” Every hand in the room goes up. And the light begins to dawn. Every one of them uses Google’s massive complex of 100,000 Linux servers, but they were blinded to the answer by a mindset in which “the software you use” is defined as the software running on the computer in front of you. Most of the “killer apps” of the Internet, applications used by hundreds of millions of people, run on Linux or FreeBSD. But the operating system, as formerly defined, is to these applications only a component of a larger system. Their true platform is the Internet.

It is in studying these next-generation applications that we can begin to understand the true long-term significance of the open source paradigm shift.

If open source pioneers are to benefit from the revolution we’ve unleashed, we must look through the foreground elements of the free and open source movements, and understand more deeply both the causes and consequences of the revolution.

Artificial intelligence pioneer Ray Kurzweil once said, “I’m an inventor. I became interested in long-term trends because an invention has to make sense in the world in which it is finished, not the world in which it is started.”[2]
I find it useful to see open source as an expression of three deep, long-term trends:

• The commoditization of software
• Network-enabled collaboration
• Software customizability (software as a service)

Long-term trends like these “three Cs,” rather than the Free Software Manifesto or The Open Source Definition, should be the lens through which we understand the changes that are being unleashed.

Software as Commodity

In his essay Some Implications of Software Commodification, Dave Stutz writes:

The word commodity is used today to represent fodder for industrial processes: things or substances that are found to be valuable as basic building blocks for many different purposes. Because of their very general value, they are typically used in large quantities and in many different ways. Commodities are always sourced by more than one producer, and consumers may substitute one producer’s product for another’s with impunity. Because commodities are fungible in this way, they are defined by uniform quality standards to which they must conform. These quality standards help to avoid adulteration, and also facilitate quick and easy valuation, which in turn fosters productivity gains.

Software commoditization has been driven by standards, in particular by the rise of communications-oriented systems such as the Internet, which depend on shared protocols, and define the interfaces and datatypes shared between cooperating components rather than the internals of those components. Such systems necessarily consist of replaceable parts. A
web server such as Apache or Microsoft’s IIS, or browsers such as Internet Explorer, Netscape Navigator®, or Mozilla, are all easily swappable, because in order to function, they must implement the HTTP protocol and the HTML data format. sendmail can be replaced by Exim or Postfix or Microsoft Exchange because all must support email exchange protocols such as SMTP, POP, and IMAP. Microsoft Outlook® can easily be replaced by Eudora®, or Pine, or Mozilla Mail, or a web mail client such as Yahoo! Mail for the same reason.

(In this regard, it’s worth noting that Unix®, the system on which Linux® is based, also has a communications-centric architecture. In The Unix Programming Environment, Kernighan and Pike eloquently describe how Unix programs should be written as small pieces designed to cooperate in “pipelines,” reading and writing ASCII files rather than proprietary data formats. Eric Raymond gives a contemporary expression of this theme in his book, The Art of Unix Programming.)

Note that in a communications-centric environment with standard protocols, both proprietary and open source software become commodities. Microsoft’s Internet Explorer web browser is just as much a commodity as the open source Apache web server, because both are constrained by the open standards of the Web. (If Microsoft had managed to gain dominant market share at both ends of the protocol pipeline between web browser and server, it would be another matter! See How the Web Was Almost Won at tim.oreilly.com for my discussion of that subject. This example makes clear one of the important roles that open source does play in “keeping standards honest.” This role is being recognized by organizations like the W3C, which are increasingly reluctant to endorse standards that have only proprietary or patent-encumbered implementations.)

What’s more, even software that starts out proprietary eventually becomes standardized and ultimately commodified. Dave
Stutz eloquently describes this process in an essay entitled *The Natural History of Software Platforms*:

> It occurs through a hardening of the external shell presented by the platform over time. As a platform succeeds in the marketplace, its APIs, UI, feature-set, file formats, and customization interfaces ossify and become more and more difficult to change. (They may, in fact, ossify so far as to literally harden into hardware appliances!) The process of ossification makes successful platforms easy targets for cloners, and cloning is what spells the beginning of the end for platform profit margins.

Consistent with this view, the cloning of Microsoft’s Windows® and Office franchises has been a major objective of the free and open source communities. In the past, Microsoft has been successful at rebuffing cloning attempts by continually revising APIs and file formats, but the writing is on the wall. Ubiquity drives standardization, and gratuitous innovation in defense of monopoly is rejected by users.

What are some of the implications of software commoditization? One might be tempted to see only the devaluation of something that was once a locus of enormous value. Thus, Red Hat® founder Bob Young once remarked, “My goal is to shrink the size of the operating system market.” (Red Hat however aimed to own a large part of that smaller market!) Defenders of the status quo, such as Microsoft VP Jim Allchin, have made statements such as “open source is an intellectual property destroyer,” and paint a bleak picture in which a great industry is destroyed, with nothing to take its place.

On the surface, Allchin appears to be right. Linux now generates tens of billions of dollars in server hardware related revenue, with the software revenues merely a rounding error. Despite Linux’s emerging dominance in the server market, Red
Hat, the largest Linux distribution company, has annual revenues of only $126 million, versus Microsoft’s $32 billion. A huge amount of software value appears to have vaporized.

But is it value or overhead? Open source advocates like to say they’re not destroying actual value, but rather squeezing inefficiencies out of the system. When competition drives down prices, efficiency and average wealth levels go up. Firms unable to adapt to the new price levels undergo what the economist E.F. Schumpeter called “creative destruction,” but what was “lost” returns manyfold as higher productivity and new opportunities.

Microsoft benefited, along with consumers, from the last round of “creative destruction” as PC hardware was commoditized. This time around, Microsoft sees the commoditization of operating systems, databases, web servers and browsers, and related software as destructive to its core business. But that destruction has created the opportunity for the killer applications of the Internet era. Yahoo!®, Google™, Amazon®, eBay®—to mention only a few—are the beneficiaries.

And so I prefer to take the view of Clayton Christensen, the author of The Innovator’s Dilemma and The Innovator’s Solution. In a recent article in Harvard Business Review, he articulates “the law of conservation of attractive profits” as follows:

When attractive profits disappear at one stage in the value chain because a product becomes modular and commoditized, the opportunity to earn attractive profits with proprietary products will usually emerge at an adjacent stage.\[3\]

We see Christensen’s thesis clearly at work in the paradigm shifts I’m discussing here.\[4\] Just as IBM’s commoditization of the basic design of the personal computer led to opportunities for attractive profits “up the stack” in software, new fortunes are being made up the stack from the commodity open source software that underlies the Internet, in a new class of proprietary applications that I have elsewhere referred to as “infoware.”
Sites such as Google, Amazon, and salesforce.com provide the most serious challenge to the traditional understanding of free and open source software. Here are applications built on top of Linux, but they are fiercely proprietary. What’s more, even when using and modifying software distributed under the most restrictive of free software licenses, the GPL, these sites are not constrained by any of its provisions, all of which are conditioned on the old paradigm. The GPL’s protections are triggered by the act of software distribution, yet web-based application vendors never distribute any software: it is simply performed on the Internet’s global stage, delivered as a service rather than as a packaged software application.

But even more importantly, even if these sites gave out their source code, users would not easily be able to create a full copy of the running application! The application is a dynamically updated database whose utility comes from its completeness and concurrency, and in many cases, from the network effect of its participating users.

(To be sure, there would be many benefits to users were some of Google’s algorithms public rather than secret, or Amazon’s 1-Click available to all, but the point remains: an instance of all of Google’s source code would not give you Google, unless you were also able to build the capability to crawl and mirror the entire Web in the same way that Google does.)

And the opportunities are not merely up the stack. There are huge proprietary opportunities hidden inside the system. Christensen notes:

> Attractive profits . . . move elsewhere in the value chain, often to subsystems from which the modular product is assembled. This is because it is improvements in the subsystems, rather than the modular product’s architecture, that drives the assembler’s ability to move upmarket towards more attractive profit margins. Hence, the subsystems become decommoditized and attractively profitable.
We saw this pattern in the PC market with most PCs now bearing the brand “Intel Inside”; the Internet could just as easily be branded “Cisco Inside”.

But these “Intel Inside” business opportunities are not always obvious, nor are they necessarily in proprietary hardware or software. The open source BIND (Berkeley Internet Name Daemon) package used to run the Domain Name System (DNS) provides an important demonstration.

The business model for most of the Internet’s commodity software turned out not to be selling that software (despite shrinkwrapped offerings from vendors such as NetManage and Spry, now long gone), but in services based on that software. Most of those businesses—the Internet Service Providers (ISPs), who essentially resell access to the TCP/IP protocol suite and to email and web servers—turned out to be low margin businesses. There was one notable exception.

BIND is probably the single most mission-critical program on the Internet, yet its maintainer has scraped by for the past two decades on donations and consulting fees. Meanwhile, domain name registration—an information service based on the software—became a business generating hundreds of millions of dollars a year, a virtual monopoly for Network Solutions, which was handed the business on government contract before anyone realized just how valuable it would be. The Intel Inside opportunity of the DNS was not a software opportunity at all, but the service of managing the namespace used by the software. By a historical accident, the business model became separated from the software.

That services based on software would be a dominant business model for open source software was recognized in The Cathedral & The Bazaar, Eric Raymond’s seminal work on the movement. But in practice, most early open source entrepreneurs focused on services associated with the maintenance
and support of the software, rather than true software as a service. (That is to say, software as a service is not service in support of software, but software in support of user-facing services!)

Dell gives us a final lesson for today’s software industry. Much as the commoditization of PC hardware drove down IBM’s outsize margins but vastly increased the size of the market, creating enormous value for users, and vast opportunities for a new ecosystem of computer manufacturers for whom the lower margins of the PC still made business sense, the commoditization of software will actually expand the software market. And as Christensen notes, in this type of market, the drivers of success “become speed to market and the ability responsively and conveniently to give customers exactly what they need, when they need it.”[5]

Following this logic, I believe that the process of building custom distributions will emerge as one of the key competitive differentiators among Linux vendors. Much as a Dell must be an arbitrageur of the various contract manufacturers vying to produce fungible components at the lowest price, a Linux vendor will need to manage the ever changing constellation of software suppliers whose asynchronous product releases provide the raw materials for Linux distributions. Companies like Debian founder Ian Murdock’s Progeny Systems already see this as the heart of their business, but even old-line Linux vendors like SuSe and new entrants like Sun tout their release engineering expertise as a competitive advantage.[6]

But even the most successful of these Linux distribution vendors will never achieve the revenues or profitability of today’s software giants like Microsoft or Oracle, unless they leverage some of the other lessons of history. As demonstrated by both the PC hardware market and the ISP industry (which as noted above is a service business built on the commodity protocols and applications of the Internet), commodity businesses are
low margin for most of the players. Unless companies find value up the stack or through an “Intel Inside” opportunity, they must compete only through speed and responsiveness, and that’s a challenging way to maintain a pricing advantage in a commodity market.

Early observers of the commodity nature of Linux, such as Red Hat’s founder Bob Young, believed that advantage was to be found in building a strong brand. That’s certainly necessary, but it’s not sufficient. It’s even possible that contract manufacturers such as Flextronix, which work behind the scenes as industry suppliers rather than branded customer-facing entities, may provide a better analogy than Dell for some Linux vendors.

In conclusion, software itself is no longer the primary locus of value in the computer industry. The commoditization of software drives value to services enabled by that software. New business models are required.

Network-Enabled Collaboration

To understand the nature of competitive advantage in the new paradigm, we should look not to Linux, but to the Internet, which has already shown signs of how the open source story will play out.

The most common version of the history of free software begins with Richard Stallman’s ethically-motivated 1984 revolt against proprietary software. It is an appealing story centered on a charismatic figure, and leads straight into a narrative in which the license he wrote—the GPL—is the centerpiece. But like most open source advocates, who tell a broader story about building better software through transparency and code sharing, I prefer to start the history with the style of software development that was normal in the early computer industry and academia. Because software was not seen as the primary source of value, source code was freely shared throughout the early computer industry.
The Unix software tradition provides a good example. Unix was developed at Bell Labs, and was shared freely with university software researchers, who contributed many of the utilities and features we take for granted today. The fact that Unix was provided under a license that later allowed ATT to shut down the party when it decided it wanted to commercialize Unix, leading ultimately to the rise of BSD Unix and Linux as free alternatives, should not blind us to the fact that the early, collaborative development preceded the adoption of an open source licensing model. Open source licensing began as an attempt to preserve a culture of sharing, and only later led to an expanded awareness of the value of that sharing.

For the roots of open source in the Unix community, you can look to the research orientation of many of the original participants. As Bill Joy noted in his keynote at the O'Reilly Open Source Convention in 1999, in science, you share your data so other people can reproduce your results. And at Berkeley, he said, we thought of ourselves as computer scientists. [7]

But perhaps even more important was the fragmented nature of the early Unix hardware market. With hundreds of competing computer architectures, the only way to distribute software was as source! No one had access to all the machines to produce the necessary binaries. (This demonstrates the aptness of another of Christensen’s “laws,” the law of conservation of modularity. Because PC hardware was standardized and modular, it was possible to concentrate value and uniqueness in software. But because Unix hardware was unique and proprietary, software had to be made more open and modular.)

This software source code exchange culture grew from its research beginnings, but it became the hallmark of a large segment of the software industry because of the rise of computer networking.
Much of the role of open source in the development of the Internet is well known: The most widely used TCP/IP protocol implementation was developed as part of Berkeley networking; BIND runs the DNS, without which none of the web sites we depend on would be reachable; sendmail is the heart of the Internet email backbone; Apache is the dominant web server; Perl the dominant language for creating dynamic sites; etc.

Less often considered is the role of Usenet in mothering the Net we now know. Much of what drove public adoption of the Internet was in fact Usenet, that vast distributed bulletin board. You “signed up” for Usenet by finding a neighbor willing to give you a newsfeed. This was a true collaborative network, where mail and news were relayed from one cooperating site to another, often taking days to travel from one end of the Net to another. Hub sites formed an ad-hoc backbone, but everything was voluntary.

Rick Adams, who created UUnet, which was the first major commercial ISP, was a free software author (though he never subscribed to any of the free software ideals—it was simply an expedient way to distribute software he wanted to use). He was the author of B News (at the time the dominant Usenet news server) as well as SLIP (Serial Line IP), the first implementation of TCP/IP for dialup lines. But more importantly for the history of the Net, Adams was also the hostmaster of the world’s largest Usenet hub. He realized that the voluntary Usenet was becoming unworkable, and that people would pay for reliable, well-connected access. UUnet started out as a non-profit, and for several years, much more of its business was based on the earlier UUCP (Unix-Unix Copy Protocol) dialup network than on TCP/IP. As the Internet caught on, UUNet and others like it helped bring the Internet to the masses. But at the end of the day, the commercial Internet industry started out of a need to provide infrastructure for the completely collaborative UUCPnet and Usenet.
The UUCPnet and Usenet were used for email (the first killer app of the Internet), but also for software distribution and collaborative tech support. When Larry Wall (later famous as the author of Perl) introduced the patch program in 1984, the ponderous process of sending around 9-track tapes of source code was replaced by the transmission of “patches”—editing scripts that update existing source files. Add in Richard Stallman’s GNU C compiler (GCC), and early source code control systems like RCS (eventually replaced by CVS and now Subversion), and you had a situation where anyone could share and update free software. The early Usenet was as much a “Napster” for shared software as it was a place for conversation.

The mechanisms that the early developers used to spread and support their work became the basis for a cultural phenomenon that reached far beyond the tech sector. The heart of that phenomenon was the use of wide-area networking technology to connect people around interests, rather than through geographical location or company affiliation. This was the beginning of a massive cultural shift that we’re still seeing today.

This cultural shift may have had its first flowering with open source software, but it is not intrinsically tied to the use of free and open source licenses and philosophies.

In 1999, together with Brian Behlendorf of the Apache project, O’Reilly founded a company called CollabNet to commercialize not the Apache product but the Apache process. Unlike many other OSS projects, Apache wasn’t founded by a single visionary developer but by a group of users who’d been abandoned by their original “vendor” (NCSA) and who agreed to work together to maintain a tool they depended on. Apache gives us lessons about intentional wide-area collaborative software development that can be applied even by companies that haven’t fully embraced open source licensing practices. For example, it is possible to apply open source collaborative principles inside a large company, even without the intention to release the resulting software to the outside world.
While CollabNet is best known for hosting high profile corporate-sponsored open source projects like OpenOffice.org, its largest customer is actually HP’s printer division, where CollabNet’s SourceCast platform is used to help more than 3000 internal developers share their code within the corporate firewall. Other customers use open-source-inspired development practices to share code with their customers or business partners, or to manage distributed worldwide development teams.

But an even more compelling story comes from that archetype of proprietary software, Microsoft. Far too few people know the story of the origin of ASP.NET. As told to me by its creators, Mark Anders and Scott Guthrie, the two of them wanted to re-engineer Microsoft’s ASP product to make it XML-aware. They were told that doing so would break backwards compatibility, and the decision was made to stick with the old architecture. But when Anders and Guthrie had a month between projects, they hacked up their vision anyway, just to see where it would go. Others within Microsoft heard about their work, found it useful, and adopted pieces of it. Some six or nine months later, they had a call from Bill Gates: “I’d like to see your project.”

In short, one of Microsoft’s flagship products was born as an internal “code fork,” the result of two developers “scratching their own itch,” and spread within Microsoft in much the same way as open source projects spread on the open Internet. It appears that open source is the “natural language” of a networked community. Given enough developers and a network to connect them, open-source-style development behavior emerges.

If you take the position that open source licensing is a means of encouraging Internet-enabled collaboration, and focus on the end rather than the means, you’ll open a much larger tent. You’ll see the threads that tie together not just traditional open source projects, but also collaborative “computing grid” projects like SETI@home, user reviews on amazon.com, technologies like collaborative filtering, new ideas about marketing such as those expressed in The Cluetrain Manifesto, weblogs,
and the way that Internet message boards can now move the stock market. What started out as a software development methodology is increasingly becoming a facet of every field, as network-enabled conversations become a principal carrier of new ideas.

I’m particularly struck by how collaboration is central to the success and differentiation of the leading Internet applications.

EBay® is an obvious example, almost the definition of a “network effects” business, in which competitive advantage is gained from the critical mass of buyers and sellers. New entrants into the auction business have a hard time competing, because there is no reason for either buyers or sellers to go to a second-tier player.

Amazon is perhaps even more interesting. Unlike eBay, whose constellation of products is provided by its users, and changes dynamically day to day, products identical to those Amazon sells are available from other vendors. Yet Amazon seems to enjoy an order-of-magnitude advantage over those other vendors. Why? Perhaps it is merely better execution, better pricing, better service, better branding. But one clear differentiator is the superior way that Amazon has leveraged its user community.

In my talks, I give a simple demonstration. I do a search for products in one of my publishing areas, JavaScript™. On , the search produces a complex page with four main areas. On the top is a block showing the three “most popular” products. Down below is a longer search listing that allows the customer to list products by criteria such as bestselling, highest-rated, by price, or simply alphabetically. On the right and the left are user-generated “ListMania” lists. These lists allow customers to share their own recommendations for other titles related to the given subject.

The section labeled “most popular” might not jump out at first. But as a vendor who sells to amazon.com, I know that it is the result of a complex, proprietary algorithm that combines not just sales but also the number and quality of user reviews,
user recommendations for alternative products, links from ListMania lists, “also bought” associations, and all the other things that Amazon refers to as the “flow” around products.

The particular search that I like to demonstrate is usually topped by my own *JavaScript: The Definitive Guide*. The book has 192 reviews, averaging 4-1/2 stars. Those reviews are among the *more than ten million* user reviews contributed by amazon.com customers.

Now contrast the #2 player in online books, barnesandnoble.com. The top result is a book published by Barnes & Noble itself, and there is no evidence of user-supplied content. *JavaScript: The Definitive Guide* has only 18 comments, the order-of-magnitude difference in user participation closely mirroring the order-of-magnitude difference in sales.

Amazon doesn’t have a natural network-effect advantage like eBay, but they’ve built one by architecting their site for user participation. Everything from user reviews, alternative product recommendations, ListMania, and the Associates program, which allows users to earn commissions for recommending books, encourages users to collaborate in enhancing the site. Amazon Web Services, introduced in 2001, take the story even further, allowing users to build alternate interfaces and specialized shopping experiences (as well as other unexpected applications) using Amazon’s data and commerce engine as a back end.

Amazon’s distance from competitors, and the security it enjoys as a market leader, is driven by the value added by its users. If, as Eric Raymond said in *The Cathedral & The Bazaar*, one of the secrets of open source is “treating your users as co-developers,” Amazon has learned this secret. But note that it’s completely independent of open source licensing practices! We start to see that what has been presented as a rigidly constrained model for open source may consist of a bundle of competencies, not all of which will always be found together.
Google makes a more subtle case for the network-effect story. Google’s initial innovation was the PageRank algorithm, which leverages the collective preferences of web users, expressed by their hyperlinks to sites, to produce better search results. In Google’s case, the user participation is extrinsic to the company and its product, and so can be copied by competitors. If this analysis is correct, Google’s long-term success will depend on finding additional ways to leverage user-created value as a key part of their offering. Services such as orkut and Gmail suggest that this lesson is not lost on them.

Now consider a counter-example. MapQuest® is another pioneer that created an innovative type of web application that almost every Internet user relies on. Yet the market is shared fairly evenly between MapQuest (now owned by AOL), maps.yahoo.com, and maps.msn.com (powered by MapPoint). All three provide a commodity-business powered by standardized software and databases. None of them have made a concerted effort to leverage user-supplied content, or engage their users in building out the application. (Note also that all three are enabling an Intel-Inside style opportunity for data suppliers such as NAVTEQ, now planning a multibillion dollar IPO!)

Continued on page 39
I've come to use the term “the architecture of participation” to describe the nature of systems that are designed for user contribution. Larry Lessig’s book, *Code and Other Laws of Cyberspace*, which he characterizes as an extended meditation on Mitch Kapor’s maxim, “architecture is politics,” made the case that we need to pay attention to the architecture of systems if we want to understand their effects.

I immediately thought of Kernighan and Pike’s description of the Unix software tools philosophy referred to above. I also recalled an unpublished portion of the interview we did with Linus Torvalds to create his essay for the 1998 book, *Open Sources*. Torvalds too expressed a sense that architecture may be more important than source code. “I couldn't do what I did with Linux for Windows, even if I had the source code. The architecture just wouldn't support it.” Too much of the windows source code consists of interdependent, tightly coupled layers for a single developer to drop in a replacement module.

And of course, the Internet and the World Wide Web have this participatory architecture in spades. As outlined above in the section on software commoditization, any system designed around communications protocols is intrinsically designed for participation. Anyone can create a participating, first-class component.

In addition, the IETF, the Internet standards process, has a great many similarities with an open source software project. The only substantial difference is that the IETF’s output is a standards document rather than a code module. Especially in the early years, anyone could participate, simply by joining a mailing list and having something to say, or by showing up to one of the three annual face-to-face meetings. Standards were decided by participating individuals, irrespective of their company affiliations. The very name for proposed Internet standards, RFCs (Request for Comments), reflects the participatory design of the Net. Though commercial participation was welcomed and encouraged, companies, like individuals, were expected to compete on the basis of their ideas and implementations, not their
money or disproportional representation. The IETF approach is where open source and open standards meet.

And while there are successful open source projects like sendmail, which are largely the creation of a single individual, and have a monolithic architecture, those that have built large development communities have done so because they have a modular architecture that allows easy participation by independent or loosely coordinated developers. The use of Perl, for example, exploded along with CPAN, the Comprehensive Perl Archive Network, and Perl’s module system, which allowed anyone to enhance the language with specialized functions, and make them available to other users.

The Web, however, took the idea of participation to a new level, because it opened that participation not just to software developers but to all users of the system.

It has always baffled and disappointed me that the open source community has not claimed the Web as one of its greatest success stories. If you asked most end users, they are most likely to associate the Web with proprietary clients such as Microsoft’s Internet Explorer than with the revolutionary open source architecture that made the Web possible. That’s a PR failure! Tim Berners-Lee’s original web implementation was not just open source, it was public domain. NCSA’s web server and Mosaic browser were not technically open source, but source was freely available. While the move of the NCSA team to Netscape sought to take key parts of the web infrastructure to the proprietary side, and the Microsoft-Netscape battles made it appear that the Web was primarily a proprietary software battleground, we should know better. Apache, the phoenix that grew from the NCSA server, kept the open vision alive, keeping the standards honest, and not succumbing to proprietary embrace-and-extend strategies.

But even more significantly, HTML, the language of web pages, opened participation to ordinary users, not just software developers. The “View Source” menu item migrated from Tim Berners-Lee’s original browser, to Mosaic, and then on to Netscape Navigator and even Microsoft’s Internet Explorer. Though no
one thinks of HTML as an open source technology, its openness was absolutely key to the explosive spread of the Web. Barriers to entry for “amateurs” were low, because anyone could look “over the shoulder” of anyone else producing a web page. Dynamic content created with interpreted languages continued the trend toward transparency.

And more germane to my argument here, the fundamental architecture of hyperlinking ensures that the value of the Web is created by its users.

In this context, it’s worth noting an observation originally made by Clay Shirky in a talk at O’Reilly’s 2001 P2P and Web Services Conference (now renamed the Emerging Technology Conference), entitled “Listening to Napster.” There are three ways to build a large database, said Clay. The first, demonstrated by Yahoo!, is to pay people to do it. The second, inspired by lessons from the open source community, is to get volunteers to perform the same task. The Open Directory Project, an open source Yahoo! competitor, is the result. (Wikipedia provides another example.) But Napster™ demonstrates a third way. Because Napster set its defaults to automatically share any music that was downloaded, every user automatically helped to build the value of the shared database.

This architectural insight may actually be more central to the success of open source than the more frequently cited appeal to volunteerism. The architecture of Linux, the Internet, and the World Wide Web are such that users pursuing their own “selfish” interests build collective value as an automatic byproduct. In other words, these technologies demonstrate some of the same network effect as eBay and Napster, simply through the way that they have been designed.

These projects can be seen to have a natural architecture of participation. But as Amazon demonstrates, by consistent effort (as well as economic incentives such as the Associates program), it is possible to overlay such an architecture on a system that would not normally seem to possess it.
Continued from page 35

**Customizability and Software-as-Service**

The last of my three Cs, customizability, is an essential concomitant of software as a service. It’s especially important to highlight this aspect because it illustrates just why dynamically typed languages like Perl, Python, and PHP, so-often denigrated by old-paradigm software developers as mere “scripting languages,” are so important on today’s software scene.

As I wrote in my 1997 essay, *Hardware, Software, and Infoware*:

If you look at a large web site like Yahoo!, you’ll see that behind the scenes, an army of administrators and programmers are continually rebuilding the product. Dynamic content isn’t just automatically generated, it is also often hand-tailored, typically using an array of quick and dirty scripting tools.

“We don’t create content at Yahoo! We aggregate it,” says Jeffrey Friedl, author of the book *Mastering Regular Expressions* and a full-time Perl programmer at Yahoo! “We have feeds from thousands of sources, each with its own format. We do massive amounts of ‘feed processing’ to clean this stuff up or to find out where to put it on Yahoo!” For example, to link appropriate news stories to tickers at *finance.yahoo.com*, Friedl needed to write a “name recognition” program able to search for more than 15,000 company names. Perl’s ability to analyze free-form text with powerful regular expressions was what made that possible.

Perl has been referred to as “the duct tape of the Internet,” and dynamic languages like Perl are important to web sites like Yahoo! and Amazon for the same reason that duct tape is important not just to heating system repairmen but to anyone who wants to hold together a rapidly changing installation. Go to any lecture or stage play, and you’ll see microphone cords and other wiring held down by duct tape.
We’re used to thinking of software as an artifact rather than a process. And to be sure, even in the new paradigm, there are software artifacts, programs, and commodity components that must be engineered to exacting specifications because they will be used again and again. But it is in the area of software that is not commoditized, the “glue” that ties together components, the scripts for managing data and machines, and all the areas that need frequent change or rapid prototyping, that dynamic languages shine.

Sites like Google, Amazon, or eBay—especially those reflecting the dynamic of user participation—are not just products, they are processes.

I like to tell people the story of the Mechanical Turk, a 1770 hoax that pretended to be a mechanical chess playing machine. The secret, of course, was that a man was hidden inside. The Turk actually played a small role in the history of computing. When Charles Babbage played against the Turk in 1820 (and lost), he saw through the hoax, but was moved to wonder whether a true computing machine would be possible.

Now, in an ironic circle, applications once more have people hidden inside them. Take a copy of Microsoft Word and a compatible computer, and it will still run ten years from now. But without the constant crawls to keep the search engine fresh, the constant product updates at an Amazon or eBay, the administrators who keep it all running, the editors and designers who integrate vendor- and user-supplied content into the interface, and in the case of some sites, even the warehouse staff who deliver the products, the Internet-era application no longer performs its function.

This is truly not the software business as it was even a decade ago. Of course, there have always been enterprise software businesses with this characteristic. (American Airlines’ Sabre reservations system is an obvious example.) But only now have they become the dominant paradigm for new computer-related businesses.
The first generation of any new technology is typically seen as an extension to the previous generations. And so, through the 1990s, most people experienced the Internet as an extension or add-on to the personal computer. Email and web browsing were powerful add-ons, to be sure, and they gave added impetus to a personal computer industry that was running out of steam.

(Open source advocates can take ironic note of the fact that many of the most important features of Microsoft’s new operating system releases since Windows 95 have been designed to emulate Internet functionality originally created by open source developers.)

But now, we’re starting to see the shape of a very different future. Napster brought us peer-to-peer file sharing, Seti@home introduced millions of people to the idea of distributed computation, and now web services are starting to make even huge database-backed sites like Amazon or Google appear to act like components of an even larger system. Vendors such as IBM and HP bandy about terms like “computing on demand” and “pervasive computing.”

The boundaries between cell phones, wirelessly connected laptops, and even consumer devices like the iPod® or TiVO®, are all blurring. Each now gets a large part of its value from software that resides elsewhere. Dave Stutz characterizes this as software above the level of a single device.\textsuperscript{[8]}

**Building the Internet Operating System**

I like to say that we’re entering the stage where we are going to treat the Internet as if it were a single virtual computer. To do that, we’ll need to create an Internet operating system.

The large question before us is this: What kind of operating system is it going to be? The lesson of Microsoft is that if you leverage insight into a new paradigm, you will find the secret
that will give you control over the industry, the “one ring to rule them all,” so to speak. Contender after contender has set out to dethrone Microsoft and take that ring from them, only to fail. But the lesson of open source and the Internet is that we can build an operating system that is designed from the ground up as “small pieces loosely joined,” with an architecture that makes it easy for anyone to participate in building the value of the system.

The values of the free and open source community are an important part of its paradigm. Just as the Copernican revolution was part of a broader social revolution that turned society away from hierarchy and received knowledge, and instead sparked a spirit of inquiry and knowledge sharing, open source is part of a communications revolution designed to maximize the free sharing of ideas expressed in code.

But free software advocates go too far when they eschew any limits on sharing, and define the movement by adherence to a restrictive set of software licensing practices. The open source movement has made a concerted effort to be more inclusive. Eric Raymond describes The Open Source Definition as a “provocation to thought,” a “social contract . . . and an invitation to join the network of those who adhere to it.”[9] But even though the open source movement is much more business friendly and supports the right of developers to choose non-free licenses, it still uses the presence of software licenses that enforce sharing as its litmus test.

The lessons of previous paradigm shifts show us a more subtle and powerful story than one that merely pits a gift culture against a monetary culture, and a community of sharers versus those who choose not to participate. Instead, we see a dynamic migration of value, in which things that were once kept for private advantage are now shared freely, and things that were once thought incidental become the locus of enormous value. It’s easy for free and open source advocates to see
this dynamic as a fall from grace, a hoarding of value that should be shared with all. But a historical view tells us that the commoditization of older technologies and the crystallization of value in new technologies is part of a process that advances the industry and creates more value for all. What is essential is to find a balance, in which we as an industry create more value than we capture as individual participants, enriching the commons that allows for further development by others.

I cannot say where things are going to end. But as Alan Kay once said, “The best way to predict the future is to invent it.”[10] Where we go next is up to all of us.

**Conclusion**

The Open Source Definition and works such as *The Cathedral & The Bazaar* tried to codify the fundamental principles of open source.

But as Kuhn notes, speaking of scientific pioneers who opened new fields of study:

Their achievement was sufficiently unprecedented to attract an enduring group of adherents away from competing modes of scientific activity. Simultaneously, it was sufficiently open ended to leave all sorts of problems for the redefined group of practitioners to resolve. Achievements that share these two characteristics, I shall refer to as “paradigms.”[11]

In short, if it is sufficiently robust an innovation to qualify as a new paradigm, the open source story is far from over, and its lessons far from completely understood. Rather than thinking of open source only as a set of software licenses and associated software development practices, we do better to think of it as a field of scientific and economic inquiry, one with many historical precedents, and part of a broader social and economic story. We must understand the impact of such factors as stan-
dards and their effect on commoditization, system architecture and network effects, and the development practices associated with software as a service. We must study these factors when they appear in proprietary software as well as when they appear in traditional open source projects. We must understand the ways in which the means by which software is deployed changes the way in which it is created and used. We must also see how the same principles that led to early source code sharing may impact other fields of collaborative activity. Only when we stop measuring open source by what activities are excluded from the definition, and begin to study its fellow travelers on the road to the future, will we understand its true impact and be fully prepared to embrace the new paradigm.

Footnotes:


2. Ray Kurzweil, speech at the Foresight Senior Associates Gathering, April 2002.


4. I have been talking and writing about the paradigm shift for years, but until I heard Christensen speak at the *Open Source Business Conference* in March 2004, I hadn’t heard his eloquent generalization of the economic principles at work in what I’d been calling business paradigm shifts. I am indebted to Christensen and to Dave Stutz, whose recent writings on software commoditization have enriched my own views on the subject.


6. From private communications with SuSe CTO Juergen Geck and Sun CTO Greg Papadopoulos.

7. I like to say that software enables speech between humans and computers. It is also the best way to talk about certain aspects of computer science, just as equations are the best way to talk about problems in physics. If you follow this line of reasoning, you realize that many of the arguments for free speech apply to open source as well. How else do you tell someone how to talk with their computer other than by sharing the code you used to do so? The benefits of open source are analogous to the benefits brought by the free flow of ideas through other forms of information dissemination.
8. Dave Stutz notes (in a private email response to an early draft of this piece), this software “includes not only what I call ‘collective software’ that is aware of groups and individuals, but also software that is customized to its location on the network, and also software that is customized to a device or a virtualized hosting environment. These additional types of customization lead away from shrinkwrap software that runs on a single PC or PDA/smartphone and towards personalized software that runs ‘on the network’ and is delivered via many devices simultaneously.”

9. From a private email response from Eric Raymond to an earlier draft of this paper.

10. Alan Kay, spoken at a 1971 internal Xerox planning meeting, as quoted at www.lisarein.com/alankay/tour.html.

Watching the “Alpha Geeks”: OS X and the Next Big Thing

This piece is based on my Keynote at the Apple Worldwide Developer Conference, May 8, 2002. I’ve added to it since then, as I’ve thought more about the issues it addresses.

We get most of our good ideas at O’Reilly by watching the “alpha geeks.” We look for people who appear to be doing magic, and ask them how they do it. (Remember Arthur C. Clarke’s dictum: “Any sufficiently advanced technology appears to be magic.”) There are always people in any field who are the most clued in to the deep trends, who seem to be playing with all the coolest stuff, and seem to have their finger in everything before most people even know about it. We get these people to tell us what they do, and we persuade them to write it down, or tell it to someone else who can write it down.

If you look at how new technologies come into play, you typically see this sequence:

1. Someone introduces a fundamental breakthrough, a disruptive technology, or business model that will change the nature of the game.

   [Aside: The term “disruptive technology” comes from Clayton Christensen’s book, *The Innovator’s Dilemma*. He cites two types of innovations: sustaining technologies (cheaper, faster, better versions of existing technologies) and disruptive technologies.]
Disruptive technologies are often not “better” when they start out—in fact, they are often worse. Case in point: the PC. It wasn’t better than the mainframe or minicomputer. It was a toy. Similarly, the World Wide Web was far less capable than proprietary CD-ROM hypertext systems and common desktop apps. And developers of both derided the Web as slow, ungainly, and ineffective. This is a typical response to disruptive technologies. Eric Raymond, speaking of open source, quoted Gandhi: “First they laugh at you, then they fight you, then you win.”

Disruptive technologies often lead to a paradigm shift (I know the phrase “paradigm shift” gets overused, but in this case, it’s spot on). The full effect of a disruptive technology paradigm shift often takes decades to be felt. There were two paradigm shifts at work in the PC revolution: first, taking the computer out of the glass house and giving it to ordinary people; and second, basing computers on commodity hardware and industry-standard designs.

There are disruptive business models as well as disruptive tech. IBM’s decision to “open source” their design and let other manufacturers copy it was critical to the growth of the market. It’s why the Intel-based PC and not the superior Apple Macintosh® became the dominant hardware platform today.

OK. So we have a disruptive innovation. What happens next?

2. Hackers and “alpha geeks” push the envelope, start to use the new technology, and get more out of their systems long before ordinary users even know what’s possible.

Both the Internet and open source were part of a hacker subculture for many years. I got my first email address back in 1978, when the ArpaNet was exactly the kind of “magic” I was talking about earlier. Some people had it. Others didn’t. (And in fact, the origins of sendmail, the mail server that
still routes the majority of Internet email, were based on exactly this disparity in skills and access. When he was a researcher at UCB, Eric Allman had ArpaNet access, and everyone wanted an account on his machine. He decided it was easier to route mail from the campus network onto the ArpaNet than to manage 700 accounts.

An example that’s still a bit far out, but that I’m confident is significant: I held a summit of peer-to-peer networking developers, and when we were sitting around having a beer afterwards, a young FreeNet developer said to Kevin Lenzo (who was there because of his early work on IRC infobots): “You sound familiar.”

Kevin mentioned that he was the developer of festvox, an open source speech synthesis package, and that he was the source of one of the voices distributed with the package. “Oh, that’s why. I listen to you all the time. I pipe IRC to festival so I can listen to it in the background when I’m coding.”

Now I’ll guarantee that lots of people will routinely be converting text to speech in a few years, and I know it because the hackers are already doing it. It’s been possible for a long time, but now it’s ripening toward the mainstream.

3. Entrepreneurs create products that simplify what the hackers came up with; there’s lots of competition around features, business model, and architecture.

A good example: On the Web, CGI was originally a hack. Then we saw a lot of different systems to improve on the CGI model, and make database-driven web sites easier for everyone: Cold Fusion, ASP, PHP, JSP.

4. Things get standardized, either by agreement or by someone winning dominant market share.

Systems get easier to use by ordinary people, but less satisfying for advanced users. During the standardization process, dominant players put up barriers to entry and try
to control the market. Entrepreneurs get acquired or squeezed out. Hackers move on to new areas, looking for “elbow room.” Innovation slows down. The cycle repeats itself.

The best platforms know how to find a balance between control and hackability, and the best companies learn how to disrupt themselves before someone else does it to them.

What Are the Alpha Geeks Telling us Right Now?

1. They are choosing Mac OS® X in overwhelming numbers!

There have been an amazing number of PowerBooks at recent O’Reilly conferences. The adoption by key open source communities and leaders is also striking. And of course, it’s not just hackers, but users who are taking up OS X in droves. Mac OS X: The Missing Manual is our fastest-selling new book since The Whole Internet User’s Guide & Catalog in 1992-1993, when the commercial Internet started to take off.

Why?

• One reason is what I call “guilt-free computing.” (Even hackers need to deal with office apps.) But everyone except the most diehard free software advocates tends to run dual-boot PCs with Linux® or FreeBSD and Windows®. And they look guiltily over their shoulder whenever they’re using Windows in public. With OS X, you get the best of both worlds in a guilt-free package! Plus Office apps, a modern GUI, and all the power of Unix development and scripting tools.

• Wireless support. The fact that 802.11b support is built-in is a huge win for hackers.
• The iApps. Apple has really built the future into this machine. iMovie®, iPhoto™, iTunes® are all very appealing built-ins.

• Sleek new packaging. The machines are just cool.

What are the hackers telling us beyond the fact that they like OS X?

2. Assume network connectivity is central; don’t treat it as an add-on. Also assume cheap local storage.

Napster™ was perhaps the first killer app that illustrated this assumption of connectivity. It may have been shut down by the legal system, but the ideas behind it are blindingly obvious in retrospect. MP3.com built massive server farms to host downloadable music archives, an approach mired in the previous generation of computing. Meanwhile, Shawn Fanning, a young student who’d grown up in the age of the Internet, asked himself, “Why do I need to have all the songs in one place? My friends already have them. All I need is a way for them to point to each other.” When everyone is connected, all that needs to be centralized is the knowledge of who has what.

We’re also seeing this mindset in projects like BitTorrent. By streaming downloads not from single sites but from a mesh of cooperating PCs, BitTorrent provides raw Internet performance increases and creates a global grid of high-performance anonymous storage.

Bob Morris, vice president at IBM Almaden, pointed out a key part of the new paradigm: storage is getting cheaper even faster than ad hoc peer-to-peer networking is. So we can assume people are on, redundant resources come and go, and find ways to navigate that paradigm. It’s really exciting to see that Apple totally gets this, with Rendezvous™
and the sharing features in iTunes. It’s also so impressive that Steve Jobs is willing to stand up to the copyright bullies and argue for the disruptive paradigm.

3. **802.11b (WiFi) networking is a must-have, not a nice-to-have.**

Some of my favorite hacker exploits are in the 802.11 space right now. Hackers are wiring their own communities. They are extending the range of 802.11b networks with homemade antennas, made from coffee cans.

And of course, Apple gets wireless. 802.11 is a key part of why hackers like Mac® laptops. The fact that it’s hidden inside the case just makes it a natural part of today’s laptop, rather than an add-on. And features like the integration of wireless with Rendezvous and iTunes are great.

4. **Chat matters.**

Long derided as a toy, chat is becoming one of the key new apps—and what’s more, key platform functionality. It’s not just chat between people, but chat between programs. As described in *Programming Jabber*, DJ Adams has been using Jabber®, the XML-based open source IM framework, to control SAP from a cell phone, using Jabber IM as the transport. Other folks are writing IM bots that answer questions and interact with users.

5. **Web Services are already a reality.**

Hackers are building unauthorized interfaces via web spidering and screen-scrapping. They think of web-facing databases as software components to be recombined in new ways, and with new interfaces. At O’Reilly, we’ve built a whole custom set of market research interfaces to Amazon by spidering every computer book on Amazon every three hours, collecting vital statistics like author, publisher, page count, price, sales rank, and number and nature of customer reviews.
Early do-it-yourself web services have tended to be inefficient, brute-force spiders, built that way because that's the only way possible. We're finally starting to see true web services from Amazon, Google, eBay®, and other sites that offer XML-based APIs. They've realized that it's in their self-interest to allow remote programmers to request the data they need, and to re-use it in creative new ways.

These four things—ad-hoc peer-to-peer networking, or Rendezvous™ (as Apple now calls it); wireless networking; chat as transport; and web sites as software components, or web services—all come together (along with a couple of other things, such as grid computing) into what I call the “emergent Internet operating system.” We really are building a next generation operating system for the Net. The question is what kind of operating system we want it to be.

I noted earlier that after the hackers push the envelope, entrepreneurs start to simplify it. Hackers have been doing the kind of things that we see in OS X for awhile, but they've had to do them the hard way. Apple has started to make it easy, so ordinary people can enjoy the benefits.

A good example is Apple’s integration of Yahoo! Finance and MapQuest right into Sherlock®, rather than just through the browser. Perl hackers have been able to do this kind of integration for years, but ordinary people couldn't enjoy it. One of the challenges, though, is not just to integrate these things into an application other than the browser, but to expose APIs so that all developers can work with them. Data-driven web sites need to be seen as software components. Google’s API is a good step in the right direction, but all data-driven web sites need to start thinking of themselves as program-callable software components for developers.

Marc Andreesen once dismissively referred to Windows as “just a bag of drivers,” but in fact, that was its genius. The
Win32 API took some of the pain out of supporting devices, so that application developers didn’t have to do lots of one-off interfaces. And one of the challenges of the “Internet OS” is to figure out the “bag of drivers” we need now. What are the services that we want to integrate that developers want to use? I believe that web databases are part of what we need standard “drivers” for.

That’s ultimately what Microsoft’s .NET is about—defining a standard set of Net-facing programming services. And Apple is showing that they have a lot of the same moxie. But since they don’t have the clout to own the whole thing, they are trying to inter-operate with the functionality that is out there. That’s one of the most exciting things about OS X: the integration of services from AOL (AIM and MapQuest) and Yahoo!. It’s absolutely the right thing to do.

What’s particularly interesting here is also the way that a non-controlling player has to do “integration into the OS.” Microsoft has typically integrated functionality that replaces or competes with the work of some existing player (Netscape, Real Networks, AOL), while Apple is having to find ways to integrate support for the existing players.

And of course, if the OS is an Internet OS rather than a PC OS, then the PC OSs are themselves software components of that larger OS. Apple seems to understand that.

So, if we’re building an Internet OS, what kind of OS would we like it to be?

**Architecture Matters**

That’s where we can learn a lot from the design of Unix and the Internet. I’m talking about design at a very high level, the “architecture is politics” level that Larry Lessig describes so
well in his book *Code and Other Laws of Cyberspace*. Both Unix and the Internet are loosely coupled protocol-centric systems rather than API-centric systems.

Kernighan and Pike’s classic, *The Unix Programming Environment*, offers a clear formulation of the few simple rules that made Unix work and laid the groundwork for what I call the architecture of participation. Use ASCII as a common data format. Write standard output and read standard input. Write small programs that work with other programs in a pipeline. This architecture made it possible for individuals all over the world to write programs without any central coordination. More than licensing, I think that architecture is behind the success of Unix, and ultimately Linux.

Similarly, the Internet was based on what Jon Postel, in the RFC for TCP, called “the robustness principle”: “Be conservative in what you do, be liberal in what you accept from others.” In other words, try to be interoperable.

The Internet architecture has flourished for 25+ years because of the robustness principle, and because it has a loosely coupled architecture in which the intelligence is in the endpoints, not in some central registry. Loosely connected actors, low barriers to entry . . . these matter.

Apple seems to be on the right track with OS X. It’s building the future into the system, in terms of the technology choices it’s making. It’s building on an open, extensible framework in the form of Darwin and FreeBSD. It’s learning lessons from the open source community.

Now, as developers, you have to do the same thing. Think network. Think open. Think extensible. Play well with others.
Hardware, Software, and Infoware

A chapter in our book Open Sources, which was published in January 1999. This is based on a talk I gave in March 1997, at the Linux Kongress in Wurzburg, the same conference where Eric Raymond first presented The Cathedral & The Bazaar. It’s an early rumination on open source and networked computing as catalysts for what I now call the Internet Operating System.

I was talking with some friends recently, friends who don’t own a computer. They were thinking of getting one so they could use Amazon.com to buy books and CDs. Not to use “the Internet,” not to use “the Web,” but to use Amazon.com.

Now, that’s the classic definition of a “killer application”: one that makes someone go out and buy a computer.

What’s interesting is that the killer application is no longer a desktop productivity application or even a back-office enterprise software system, but an individual web site. And once you start thinking of web sites as applications, you soon come to realize that they represent an entirely new breed, something you might call an “information application,” or perhaps even “infoware.”

Information applications are used to computerize tasks that just couldn’t be handled in the old computing model. A few years ago, if you wanted to search a database of a million books, you talked to a librarian, who knew the arcane search syntax of the available computerized search tools and might
be able to find what you wanted. If you wanted to buy a book, you went to a bookstore, and looked through its relatively small selection. Now, tens of thousands of people with no specialized training find and buy books online from that million-record database every day.

The secret is that computers have come one step closer to the way that people communicate with each other. Web-based applications use plain English to build their interface—words and pictures, not specialized little controls that acquire meaning only as you learn the software.

Traditional software embeds small amounts of information in a lot of software; infoware embeds small amounts of software in a lot of information. The “actions” in an infoware product are generally fairly simple: make a choice, buy or sell, enter a small amount of data, and get back a customized result.

These actions are often accomplished by scripts attached to a hypertext link using an interface specification called CGI (the Common Gateway Interface). CGI defines a way for a web server to call any external program and return the output of that program as a web page. CGI programs may simply be small scripts that perform a simple calculation, or they may connect to a full-fledged back-end database server. But even when there’s a heavy-duty software engine behind a site, the user interface itself is not composed of traditional software. The interface consists of web pages (which may well have been created by a writer, editor, or designer rather than by a programmer).

Information interfaces are typically dynamic. For example, Amazon.com’s presentation of books is driven by sales rankings that are updated every hour. Customers can add comments and ratings on the fly, which then become a key part of the information-rich decision-support interface for purchasers. A site designed to help someone buy or sell stocks online needs to not only present updated share prices, but also
the latest relevant news stories, insider trading information, analyst recommendations, and perhaps even user discussion groups. The information interface thus typically consists of a rich mix of hand-crafted documents, program-generated data, and links to specialized application servers (such as email, chat, or conferencing).

Information interfaces are not as efficient for tasks that you do over and over as pure software interfaces, but they are far better for tasks you do only rarely, or differently each time. In particular, they are good for interfaces in which you make choices based on information presented to you. Whether you're buying a book or CD at Amazon.com, or a stock at E*Trade, the actual purchase is a fairly trivial part of the interaction. It’s the quality of the information provided to help you make a decision that forms the heart of the application you interact with.

The way the Web is transforming the whole computing paradigm was never clearer to me than back in 1994, before Microsoft had gotten the web religion, and I shared the stage (via satellite) with Microsoft VP Craig Mundie at an NTT event in Japan. Mundie was demonstrating the planned interface for Microsoft’s “Tiger” server, which was supposed to enable video on demand. The interface emulated Windows, with cascading menus responding to a kind of virtual remote control channel clicker.

It was pretty obvious to those of us who were involved in the Web that the right interface for video on demand, when and if it comes, will be a web-like information interface. It’s ironic that even then, Microsoft had the perfect interface for video-on-demand: its own CD-ROM-based movie encyclopedia, Cinemania®. What better way to choose what movie to watch than to search by category, read a few reviews, watch a few film clips, and then, homework done, click on a hypertext link to start the movie? Cinemania has it all but the last step. It’s not until hypertext-based information products are connected to
network servers that their real power becomes apparent. Suddenly, information is not an end in itself, but an interface that allows a user to control an application space far too complex for a traditional software application. (Amazon.com clearly knows this: their purchase of the Internet Movie Database, a collection of user reviews and other information about movies, will put them in pole position not only for selling videotapes online, but as a future gateway for video-on-demand services.)

Information interfaces are particularly appropriate for decision-support applications, but they also make sense for one-time tasks. In a sense, the use of “wizards” for software installation is an example of the same trend.

There are also information applications that use a simpler, more software-like interface for user interaction, but provide dynamic information output. My favorite example is something that would have been virtually unthinkable as an application only a few years ago: getting maps and directions. A mapping site like maps.yahoo.com lets you type in two addresses, and get back a map and a set of directions showing how to get to one from the other.

So what does all this have to do with open source software?

There’s an obvious answer: most of the technologies that make the Web possible are open source.

The Internet itself—features like the TCP/IP network protocol and key infrastructure elements such as the Domain Name System (DNS) were developed through the open-source process. It’s easy to argue that the open source BIND (Berkeley Internet Name Daemon) program that runs the DNS is the single most mission-critical Internet application. Even though most web browsing is done with proprietary products (Netscape’s Navigator and Microsoft’s Internet Explorer), both are outgrowths of Tim Berners-Lee’s original open source web
implementation and open protocol specification. According to the automated Netcraft web server survey (www.netcraft.co.uk/survey), more than 50% of all visible web sites are served by the open source Apache web server. The majority of web-based dynamic content is generated by open source scripting languages such as Perl, Python, and Tcl.

But this obvious answer is only part of the story. After all, why is it the Web and not some proprietary technology that is the basis for the networked information applications of the future?

Microsoft actually was ahead of the curve in realizing the power of online multimedia. In 1994, when the Web started to take off, Microsoft’s CD-ROM products like Encarta®, their online encyclopedia, and Cinemania, their online movie reference, were ahead of the Web in providing online hyperlinked documents with rich multimedia capabilities. Microsoft even realized that it was important to provide information resources via online networks.

There was only one problem with Microsoft’s vision of the Microsoft Network: barriers to entry were high. Publishers were expected to use proprietary Microsoft tools, to apply and be approved by Microsoft, and to pay to play. By contrast, anyone could start a web site. The software you needed was free. The specifications for creating documents and dynamic content were simple, open, and clearly documented.

Perhaps even more important, both the technology and the Internet ethic made it legitimate to copy features from other people’s web sites. The HTML (HyperText Markup Language) pages that were used to implement various features on a web site could be easily saved and imitated. Even the CGI scripts used to create dynamic content were available for copying. Although traditional computer languages like C run faster, Perl became the dominant language for CGI because it was more accessible. While Perl is powerful enough to write major
applications, it is possible for amateurs to write small scripts to accomplish specialized tasks. Even more important, because Perl is not a compiled language, the scripts that are used on web pages can be viewed, copied, and modified by users. In addition, archives of useful Perl scripts were set up and freely shared among web developers. The easy cloning of web sites built with the HTML+CGI+Perl combination meant that for the first time, powerful applications could be created by non-programmers.

In this regard, it’s interesting to point out that the software industry’s first attempts to improve on the web interface for active content—technologies like browser-side Java™ applets and Microsoft ActiveX® controls—failed because they were aimed at professional programmers and could not easily be copied and implemented by the amateurs who were building the Web. Vendors viewed the Web in software terms, and didn’t understand that the Web was changing not only what applications were being built but what tools their builders needed.

Industry analysts have been predicting for years that Perl and CGI will be eclipsed by newer software technologies. But even now, when major web sites employ large staffs of professional programmers, and newer technologies like Microsoft’s Active Server Pages (ASP) and Sun’s Java servlets are supplanting CGI for performance reasons, Perl continues to grow in popularity. Perl and other open source scripting languages such as Python and Tcl remain central to web sites large and small because infoware applications are fundamentally different than software applications and require different tools.

If you look at a large web site like Yahoo!, you’ll see that behind the scenes, an army of administrators and programmers are continually rebuilding the product. Dynamic content isn’t just automatically generated, it is also often hand-tailored, typically using an array of quick and dirty scripting tools.
“We don’t create content at Yahoo! We aggregate it,” says Jeffrey Friedl, author of the book Mastering Regular Expressions and a full-time Perl programmer at Yahoo!. “We have feeds from thousands of sources, each with its own format. We do massive amounts of ‘feed processing’ to clean this stuff up or to find out where to put it on Yahoo!.” For example, to link appropriate news stories to tickers at quotes.yahoo.com, Friedl needed to write a “name recognition” program able to search for more than 15,000 company names. Perl’s ability to analyze free-form text with powerful regular expressions was what made that possible.

Perl is also a central component in the system administration infrastructure used to keep the site live and current. Vast numbers of Perl scripts are continually crawling the Yahoo! servers and their links to external sites, and paging the staff whenever a URL doesn’t return the expected result. The best-known of these crawlers is referred to as “the Grim Reaper.” If an automated connection to a URL fails more than the specified number of times, the page is removed from the Yahoo! directory.

Amazon.com is also a heavy user of Perl. The Amazon.com authoring environment demonstrates Perl’s power to tie together disparate computing tools; it’s a “glue language” par excellence. A user creates a new document with a form that calls up a Perl program, which generates a partially-completed SGML document, then launches either Microsoft Word or GNU Emacs (at the user’s choice), but also integrates CVS (Concurrent Versioning System) and Amazon.com’s home-grown SGML tools. The Amazon.com SGML classes are used to render different sections of the web site—for example, HTML with or without graphics—from the same source base. A Perl-based parser renders the SGML into HTML for approval before the author commits the changes.

Perl has been called “the duct tape of the Internet,” and like duct tape, it is used in all kinds of unexpected ways. Like a
movie set held together with duct tape, a web site is often put up and torn down in a day, and needs lightweight tools and quick but effective solutions.

Microsoft’s failed attempt to turn infoware back into software with ActiveX® is rooted in the way paradigms typically shift in the computer industry. As a particular market segment matures, the existing players have an enormous vested interest in things continuing the way they are. This makes it difficult for them to embrace anything really new, and allows—almost requires—that new players (“the barbarians,” to use Philippe Kahn’s phrase) come in to create the new markets.

Microsoft’s ascendancy over IBM as the ruling power of the computer industry is a classic example of how this happened the last time around. IBM gave away the market to Microsoft because it didn’t see that the shift of power was not only from the glass house to the desktop, but also from proprietary to commodity hardware and from hardware to software.

In the same way, despite its attempts to get into various information businesses, Microsoft still doesn’t realize—perhaps can’t realize and still be Microsoft—that software, as Microsoft has known it, is no longer the central driver of value creation in the computer business.

In the days of IBM’s dominance, hardware was king, and the barriers to entry into the computer business were high. Most software was created by the hardware vendors, or by software vendors who were satellite to them.

The availability of the PC as a commodity platform (as well as the development of open systems platforms such as Unix) changed the rules in a fundamental way. Suddenly, the barriers to entry were low, and entrepreneurs such as Mitch Kapor of Lotus and Bill Gates took off.

If you look at the early history of the Web, you see a similar pattern. Microsoft’s monopoly on desktop software had made
the barriers to entry in the software business punishingly high. What’s more, software applications had become increasingly complex, with Microsoft putting up deliberate barriers to entry against competitors. It was no longer possible for a single programmer in a garage (or a garret) to make an impact.

This is perhaps the most important point to make about open source software: it lowers the barriers to entry into the software market. You can try a new product for free—and even more than that, you can build your own custom version of it, also for free. Source code is available for massive independent peer review. If someone doesn’t like a feature, they can add to it, subtract from it, or reimplement it. If they give their fix back to the community, it can be adopted widely very quickly.

What’s more, because developers (at least initially) aren’t trying to compete on the business end, but instead focus simply on solving real problems, there is room for experimentation in a less punishing environment. As has often been said, open source software “lets you scratch your own itch.” Because of the distributed development paradigm, with new features being added by users, open source programs “evolve” as much as they are designed.

Indeed, the evolutionary forces of the market are freer to operate as nature “intended” when unencumbered by marketing barriers or bundling deals, the equivalent of prosthetic devices that help the less-than-fit survive.

*Evolution breeds not a single winner, but diversity.*

It is precisely the idiosyncratic nature of many of the open source programs that is their greatest strength. Again, it’s instructive to look at the reasons for Perl’s success.

Larry Wall originally created Perl to automate some repetitive system administration tasks he was faced with. After releasing the software to the Net, he found more and more applications, and the language grew, often in unexpected directions.
Perl has been described as a “kitchen sink language” because its features seem chaotic to the designers of more “orthogonal” computer languages. But chaos can often reveal rich structure. Chaos may also be required to model what is inherently complex. Human languages are complex because they model reality. As Wall says, “English is useful because it’s a mess. Since English is a mess, it maps well onto the problem space, which is also a mess. . . . Similarly, Perl was designed to be a mess (though in the nicest of possible ways).”

The open source development paradigm is an incredibly efficient way of getting developers to work on features that matter. New software is developed in a tight feedback loop with customer demand, without distortions caused by marketing clout or top-down purchasing decisions. Bottom-up software development is ideal for solving bottom-up problems.

Using the open source software at the heart of the Web, and its simpler development paradigm, entrepreneurs like Jerry Yang and David Filo were able to do just that. It’s no accident that Yahoo!, the world’s largest and most successful web site, is built around freely available open source software: the FreeBSD operating system, Apache, and Perl.

Just as it was last time around, the key to the next stage of the computer industry is in fact the commoditization of the previous stage. As Bob Young of Red Hat®, the leading Linux® distributor, has noted, his goal is not to dethrone Microsoft at the top of the operating systems heap, but rather, to shrink the dollar value of the operating systems market.

The point is that open source software doesn’t need to beat Microsoft at its own game. Instead it is changing the nature of the game.

To be sure, for all their astronomical market capitalization, information-application providers such as Amazon.com and Yahoo! are still tiny compared to Microsoft. But the writing on
the wall is clear. The edges of human-computer interaction, the opportunities for computerizing tasks that haven't been computerized before, are in infoware, not in software.

As the new “killer applications” emerge, the role of software will increasingly be as an enabler for infoware. There are enormous commercial opportunities to provide web servers, database backends and application servers, and network programming languages like Java, as long as these products fit themselves into the new model rather than trying to supplant it. Note that in the shift from a hardware-centric to a software-centric computer industry, hardware didn’t go away. IBM still flourishes as a company (though most of its peers have downsized or capsized). But other hardware players emerged who were suited to the new rules: Dell, Compaq, and especially Intel.

Intel realized that the real opportunity for them was not in winning the computer systems wars, but in being an arms supplier to the combatants.

The real challenge for open source software is not whether it will replace Microsoft in dominating the desktop, but rather whether it can craft a business model that will help it to become the “Intel Inside” of the next generation of computer applications.

Otherwise, the open source pioneers will be shouldered aside just as Digital Research was in the PC operating system business by someone who understands precisely where the current opportunity lies.

But however that turns out, open source software has already created a fork in the road. Just as the early microcomputer pioneers (in both hardware and software) set the stage for today’s industry, open source software has set the stage for the drama that is just now unfolding, and that will lead to a radical reshaping of the computer industry landscape over the next five to ten years.
On Publishing
Piracy is Progressive Taxation
and Six Other Lessons on the Evolution of Online Distribution

December 2002

The continuing controversy over online file sharing sparks me to offer a few thoughts as an author and publisher. To be sure, I write and publish neither movies nor music, but books. But I think that some of the lessons of my experience still apply.

Lesson 1

Obscurity is a Far Greater Threat to Authors and Creative Artists than Piracy.

Let me start with book publishing. More than 100,000 books are published each year, with several million books in print, yet fewer than 10,000 of those new books have any significant sales, and only a hundred thousand or so of all the books in print are carried in even the largest stores. Most books have a few months on the shelves of the major chains, and then wait in the darkness of warehouses from which they will move only to the recycling bin. Authors think that getting a publisher will be the realization of their dreams, but for so many, it’s just the start of a long disappointment.

Sites like Amazon that create a virtual storefront for all the books in print cast a ray of light into the gloom of those warehouses, and so books that would otherwise have no outlet at all can be discovered and bought. Authors who are fortunate enough to get the rights to their books back from the publisher often put them up freely online, in hopes of finding
readers. The Web has been a boon for readers, since it makes it easier to spread book recommendations and to purchase the books once you hear about them. But even then, few books survive their first year or two in print. Empty the warehouses and you couldn’t give many of them away.

Many works linger in deserved obscurity, but so many more suffer simply from the vast differential between supply and demand.

I don’t know the exact size of the entire CD catalog, but I imagine that it is similar in scope. Tens of thousands of musicians self-publish their own CDs; a happy few get recording contracts. Of those, fewer still have their records sell in appreciable numbers. The deep backlist of music publishers is lost to consumers because the music just isn’t available in stores.

There are fewer films, to be sure, because of the cost of film making, but even there, obscurity is a constant enemy. Thousands of independent film makers are desperate for distribution. A few independent films, like Denmark’s Dogme films, get visibility. But for most, visibility is limited to occasional showings at local film festivals. The rise of digital video also promises that film making will soon be as much a garage opportunity as starting a rock band, and as much of a garret opportunity as the great American novel.

Lesson 2

Piracy is Progressive Taxation.

For all of these creative artists, most laboring in obscurity, being well-enough known to be pirated would be a crowning achievement. Piracy is a kind of progressive taxation, which may shave a few percentage points off the sales of well-known artists (and I say “may” because even that point is not proven), in exchange for massive benefits to the far greater number for whom exposure may lead to increased revenues.
Our current distribution systems for books, music, and movies are skewed heavily in favor of the “haves” against the “have nots.” A few high-profile products receive the bulk of the promotional budget and are distributed in large quantities; the majority depend, in the words of Tennessee Williams’ character Blanche DuBois, “on the kindness of strangers.”

Lowering the barriers to entry in distribution, and the continuous availability of the entire catalog rather than just the most popular works, is good for artists, since it gives them a chance to build their own reputation and visibility, working with entrepreneurs of the new medium who will be the publishers and distributors of tomorrow.

I have watched my 19 year-old daughter and her friends sample countless bands on Napster and Kazaa and, enthusiastic for their music, go out to purchase CDs. My daughter now owns more CDs than I have collected in a lifetime of less exploratory listening. What’s more, she has introduced me to her favorite music, and I too have bought CDs as a result. And no, she isn’t downloading Britney Spears, but forgotten bands from the 60s, 70s, 80s, and 90s, as well as their musical forebears in other genres. This is music that is difficult to find—except online—but, once found, leads to a focused search for CDs, records, and other artifacts. eBay is doing a nice business with much of this material, even if the RIAA fails to see the opportunity.

Lesson 3

Customers Want to Do the Right Thing, if They Can.

Piracy is a loaded word, which we used to reserve for wholesale copying and resale of illegitimate product. The music and film industry usage, applying it to peer-to-peer file sharing, is a disservice to honest discussion.
Online file sharing is the work of enthusiasts who are trading their music because there is no legitimate alternative. Piracy is an illegal commercial activity that is typically a substantial problem only in countries without strong enforcement of existing copyright law.

At O’Reilly, we publish many of our books in online form. There are people who take advantage of that fact to redistribute unpaid copies. (The biggest problem, incidentally, is not on file sharing networks, but from copies of our CD Bookshelf product line being put up on public web servers, or copied wholesale and offered for sale on eBay.) While these pirated copies are annoying, they hardly destroy our business. We’ve found little or no abatement of sales of printed books that are also available for sale online.

What’s more, many of those who do infringe respond to little more than a polite letter asking them to take the materials down. Those servers that ignore our requests are typically in countries where the books are not available for sale or are far too expensive for local consumers to buy.

What’s even more interesting, though, is that our enforcement activities are customer-driven. We receive thousands of emails from customers letting us know about infringing copies and sites. Why? They value our company and our authors, and they want to see our work continue. They know that there is a legitimate way to pay for online access—our Safari® Books Online subscription service (safari.oreilly.com) can be had for as little as $9.99 a month—and accordingly recognize free copies as illegitimate.

A similar data point comes from Jon Schull, the former CTO of Softlock, the company that worked with Stephen King on his eBook experiment, Riding the Bullet. Softlock, which used a strong DRM scheme, was relying on “superdistribution” to reduce the costs of hosting the content—the idea that customers would redistribute their copies to friends, who would
then simply need to download a key to unlock said copy. But most of the copies were downloaded anyway and very few were passed along. Softlock ran a customer survey to find out why there was so little “pass-along” activity. The answer, surprisingly, was that customers didn’t understand that redistribution was desired. They didn’t do it because they “thought it was wrong.”

The simplest way to get customers to stop trading illicit digital copies of music and movies is to give those customers a legitimate alternative, at a fair price.

Lesson 4

Shoplifting is a Bigger Threat than Piracy.

While few of the people putting books on public web servers seek to profit from the activity, those who are putting up CDs for sale on eBay containing PDF or HTML copies of dozens of books are in fact practicing piracy—organized copying of content for resale.

But even so, we see no need for stronger copyright laws, or strong Digital Rights Management software, because existing law allows us to prosecute the few deliberate pirates.

We don’t have a substantial piracy problem in the U.S. and Europe. The fact that its software products have been available for years on Warez sites (and now on file trading networks) has not kept Microsoft from becoming one of the world’s largest and most successful companies. Estimates of “lost” revenue assume that illicit copies would have been paid for; meanwhile, there is no credit on the other side of the ledger for copies that are sold because of “upgrades” from familiarity bred by illicit copies.

What we have is a problem that is analogous, at best, to shoplifting, an annoying cost of doing business.
And overall, as a book publisher who also makes many of our books available in electronic form, we rate the piracy problem as somewhere below shoplifting as a tax on our revenues. Consistent with my observation that obscurity is a greater danger than piracy, shoplifting of a single copy can lead to lost sales of many more. If a bookstore has only one copy of your book, or a music store one copy of your CD, a shoplifted copy essentially makes it disappear from the next potential buyer’s field of possibility. Because the store’s inventory control system says the product hasn’t been sold, it may not be reordered for weeks or months, perhaps not at all.

I have many times asked a bookstore why they didn’t have copies of one of my books, only to be told, after a quick look at the inventory control system: “But we do. It says we still have one copy in stock, and it hasn’t sold in months, so we see no need to reorder.” It takes some prodding to force the point that perhaps it hasn’t sold because it is no longer on the shelf.

Because an online copy is never out of stock, we at least have a chance at a sale, rather than being subject to the enormous inefficiencies and arbitrary choke points in the distribution system.

Lesson 5

**File Sharing Networks Don’t Threaten Book, Music, or Film Publishing. They Threaten Existing Publishers.**

The music and film industries like to suggest that file sharing networks will destroy their industries.

Those who make this argument completely fail to understand the nature of publishing. Publishing is not a role that will be undone by any new technology, since its existence is mandated by mathematics. Millions of buyers and millions of sellers cannot find one another without one or more middlemen
who, like a kind of step-down transformer, segment the market into more manageable pieces. In fact, there is usually a rich ecology of middlemen. Publishers aggregate authors for retailers. Retailers aggregate customers for publishers. Wholesalers aggregate small publishers for retailers and small retailers for publishers. Specialty distributors find ways into non-standard channels.

Those of us who watched the rise of the Web as a new medium for publishing have seen this ecology evolve within less than a decade. In the Web’s early days, rhetoric claimed that we faced an age of disintermediation, that everyone could be his or her own publisher. But before long, individual web site owners were paying others to help them increase their visibility in Yahoo!, Google, and other search engines (the equivalent of Barnes & Noble and Borders for the Web), and web authors were happily writing for sites like AOL and MSN, or on the technology side, CNET, Slashdot, O’Reilly Network, and other web publishers. Meanwhile, authors from Matt Drudge to Dave Winer and Cory Doctorow made their names by publishing for the new medium.

As Jared Diamond points out in his book *Guns, Germs, and Steel*, mathematics is behind the rise of all complex social organization.

There is nothing in technology that changes the fundamental dynamic by which millions of potentially fungible products reach millions of potential consumers. The means by which aggregation and selection are made may change with technology, but the need for aggregation and selection will not. Google’s use of implicit peer recommendation in its page rankings plays much the same role as the large retailers’ use of detailed sell-through data to help them select their offerings.

The question before us is not whether technologies such as peer-to-peer file sharing will undermine the role of the creative artist or the publisher, but how creative artists can leverage
new technologies to increase the visibility of their work. For publishers, the question is whether they will understand how to perform their role in the new medium before someone else does. Publishing is an ecological niche; new publishers will rush in to fill it if the old ones fail to do so.

If we take the discussion back to first principles, we understand that publishing isn’t just about physical aggregation of product but also requires an intangible aggregation and management of “reputation.” People go to Google or Yahoo!, Barnes & Noble or Borders, HMV, or MediaPlay, because they believe that they will find what they want there. And they seek out particular publishers, like Knopf or O’Reilly, because we have built a track-record of trust in our ability to find interesting topics and skilled authors.

Now, let’s take this discussion over to music file sharing. How do people find songs on Kazaa or any of the other post-Napster file sharing services? First, they may be looking for a song they already know. But such searches for a known artist or song title are fundamentally self-limiting, since they depend on the marketing of a “name space” (artist/song pairs) that is extrinsic to the file sharing service. To truly supplant the existing music distribution system, any replacement must develop its own mechanisms for marketing and recommendation of new music.

And in fact, we already see those mechanisms emerging. File sharing services rely heavily on that most effective of marketing techniques: word of mouth. But over time, anyone who has studied the evolution of previous media will see that searches based on either pre-existing knowledge or word of mouth represent only the low-hanging fruit. As the market matures, paid marketing is added, and step by step, we build up the same rich ecology of middlemen that characterizes existing media marketplaces.
New media have historically not replaced but rather augmented and expanded existing media marketplaces, at least in the short term. Opportunities exist to arbitrage between the new distribution medium and the old, as, for instance, the rise of file sharing networks has helped to fuel the trading of records and CDs (unavailable through normal recording industry channels) on eBay.

Over time, it may be that online music publishing services will replace CDs and other physical distribution media, much as recorded music relegated sheet music publishers to a niche and, for many, made household pianos a nostalgic affectation rather than the home entertainment center. But the role of the artist and the music publisher will remain. The question then, is not the death of book publishing, music publishing, or film production, but rather one of who will be the publishers.

Lesson 6

“Free” is Eventually Replaced by a Higher-Quality Paid Service.

A question for my readers: How many of you still get your email via peer-to-peer UUCP dialups or the old “free” Internet, and how many of you pay $19.95 a month or more to an ISP? How many of you watch “free” television over the airwaves, and how many of you pay $20-$60 a month for cable or satellite television? (Not to mention continue to rent movies on videotape and DVD, and purchasing physical copies of your favorites.)

Services like Kazaa flourish in the absence of competitive alternatives. I confidently predict that once the music industry provides a service that provides access to all the same songs, freedom from onerous copy-restriction, more accurate metadata and other added value, there will be hundreds of millions of paying subscribers. That is, unless they wait too long, in which case, Kazaa itself will start to offer (and charge for)
these advantages. (Or would, in the absence of legal challenges.) Much as AOL, MSN, Yahoo!, CNET, and many others have collectively built a multibillion dollar media business on the “free” Web, “publishers” will evolve on file sharing networks.

Why would you pay for a song that you could get for free? For the same reason that you will buy a book that you could borrow from the public library or buy a DVD of a movie that you could watch on television or rent for the weekend. Convenience, ease-of-use, selection, ability to find what you want, and for enthusiasts, the sheer pleasure of owning something you treasure.

The current experience of online file sharing services is mediocre at best. Students and others with time on their hands may find them adequate. But they leave much to be desired, with redundant copies of uneven quality, intermittent availability of some works, incorrect identification of artist or song, and many other quality problems.

Opponents may argue that the Web demonstrates precisely what they are afraid of, that content on the Web is “free,” that advertising is an insufficient revenue model for content providers, and that subscription models have not been successful. However, I will argue that the story is still unfinished.

Subscription sites are on the rise. Computer industry professionals can be seen as the “early adopters” in this market. For example, O’Reilly’s Safari Books Online is growing at 30 percent a month, and now represents a multimillion dollar revenue stream for us and other participating publishers.

Most observers also seem to miss the point that the Internet is already sold as a subscription service. All we’re working on is the development of added-value premium services. What’s more, there are already a few vertically-integrated ISPs (notably AOL Time Warner) that provide “basic” connectivity but own vast libraries of premium content.
In looking at online content subscription services, analogies with television are instructive. Free, advertiser-supported television has largely been supplanted—or should I say supplemented (because the advertising remains)—by paid subscriptions to cable TV. What’s more, revenue from “basic cable” has been supplemented by various aggregated premium channels. HBO, one of those channels, is now television’s most profitable network. Meanwhile, over on the Internet, people pay their ISP $19.95/month for the equivalent of “basic cable,” and an ideal opportunity for a premium channel, a music download service, has gone begging for lack of vision on the part of existing music publishers.

Another lesson from television is that people prefer subscriptions to pay-per-view, except for very special events. What’s more, they prefer subscriptions to larger collections of content, rather than single channels. So, people subscribe to “the movie package,” “the sports package,” and so on. The recording industry’s “per song” trial balloons may work, but I predict that in the long term, an “all-you-can-eat” monthly subscription service (perhaps segmented by musical genre) will prevail in the marketplace.

Lesson 7

There’s More Than One Way To Do It.

A study of other media marketplaces shows, though, that there is no single silver-bullet solution. A smart company maximizes revenue through all its channels, realizing that its real opportunity comes when it serves the customer whom ultimately pays its bills.

At O’Reilly, we’ve been experimenting with online distribution of our books for years. We know that we must offer a compelling online alternative before someone else does. As the
Hawaiian proverb says, “No one promised us tomorrow.” Competition with free alternatives forces us to explore new distribution media and new forms of publishing.

In addition to the Safari subscription service mentioned above, we publish an extensive network of advertising-supported “free” information sites as the O’Reilly Network (www.oreillynet.com). We have published a number of books under “open publication licenses” where free redistribution is explicitly allowed (www.oreilly.com/openbook). We do this for several reasons: to build awareness of products that might otherwise be ignored; to build brand loyalty among online communities; or, sometimes, because a product can no longer be economically sold in traditional channels, and we’d rather make it available for free than have it completely disappear from the market.

We have also published many of our books on CD-ROM, in a format referred to as the CD Bookshelf, typically a collection of a half dozen or so related books.

And of course, we continue to publish print books. The availability of free online copies is sometimes used to promote a topic or author (as books such as The Cathedral & The Bazaar or The Cluetrain Manifesto became bestsellers in print as a result of the wide exposure it received online). We make available substantial portions of all of our books online, as a way for potential readers to sample what they contain. We’ve even found ways to integrate our books into the online help system for software products, including Dreamweaver® and Microsoft’s Visual Studio®.

Interestingly, some of our most successful print/online hybrids have come about where we present the same material in different ways for the print and online contexts. For example, much of the content of our bestselling book Programming Perl (more than 600,000 copies in print) is available online as part
of the standard Perl documentation. But the entire package—not to mention the convenience of a paper copy, and the aesthetic pleasure of the strongly branded packaging—is only available in print. Multiple ways to present the same information and the same product increase the overall size and richness of the market.

And that’s the ultimate lesson. “Let the Wookiee win!” as C3PO said so memorably in the first *Star Wars* movie. Let him win in as many ways as you can find, at a fair price, and let him choose which works best for him.
Information Wants to be Valuable

May 2001

Originally written for a Nature magazine series on the impact of the Web on publishing, and still available at www.nature.com/nature/debates/e-access/index.html. I argue that the particulars of the publisher’s role certainly change when works are published online, but the value of their work to both authors and readers remains constant.

“Information doesn’t want to be free. Information wants to be valuable.” I first heard this gem from Larry Wall, creator of the Perl programming language. Like many other open source software authors, from Linus Torvalds, creator of Linux®, to Tim Berners-Lee and his spiritual descendants at the Apache web server project, Larry discovered that one way to make his information (i.e., his software) more valuable was to make it free. Larry was thus able to increase its utility not only for himself (because others who took it up made changes and enhancements that he could use), but for everyone else who uses it, because as software becomes more ubiquitous it can be taken for granted as a foundation for further work. The Internet (based on freely available software including TCP/IP, BIND, Apache, sendmail, and so on) demonstrates clearly just how much value can be created by the distribution of freely available software.
Nonetheless, it is also clear that others, Bill Gates being the paramount example, have found that the best way to make their information valuable is to restrict access to it. No one can question that Microsoft has created enormous value for itself and its shareholders, and even its critics should admit that Microsoft has been a key enabler of the ubiquitous personal computing on which so much of our modern business world depends.

What many people fail to realize is that both Larry Wall and Bill Gates have a great deal in common: as the creators (albeit with a host of co-contributors) of a body of intellectual work, they have made strategic decisions about how best to maximize its value. History has proven that each of their strategies can work. The question, then, is one of goals, and of the strategies to reach those goals. The question for publishers and other middlemen who are not themselves the creators of the content they distribute, is how best to serve those goals. Information wants to be valuable. Publishers must focus on increasing the value, both to its producers and to its consumers, of the information they aggregate and distribute.

I am neither a practicing scientist nor a publisher of scientific journals, but as a book and web publisher who works on a regular basis to document widely available “infrastructure” software (both free and commercial), I am daily confronted with decisions akin to those reflected in the debate now being carried in these pages. Because I publish books about free software, the people best qualified to write about it are often the authors of the software. Like scientists, those authors often have as their ideal the widest possible dissemination of their software and information about how to use it, rather than the greatest economic gain. They would like to see the documentation they write distributed freely along with the software.

At other times, though, software authors see documentation as an afterthought. They would rather not deal with it, and hope that someone else will. In those cases, the question of com-
pensation often comes into play. Will a third party who is motivated chiefly by money earn enough from this book to justify the time writing it?

In helping authors to navigate this discussion, I try to bring them back to their goal. Is it maximum dissemination of information or is it earning enough to justify the work? I should note that the jury is still out on whether making the text of a book freely available helps or hurts sales of a print book. There is evidence on both sides.

In some cases, such as Eric Raymond’s book, The Cathedral & The Bazaar, free distribution of the content created the “buzz” that allowed us to publish the same material successfully in print. In other cases, such as our initial publication of the Linux Network Administrator’s Guide, sales were reduced because other companies republished some or all of the book at lower cost, which they could do because they had no development costs or royalties. However, over time this problem abated, because the fact that those publishers were not adding value was recognized by the target audience, and eventually marginalized their products.

I see many parallels between the work of free software authors and the work of scientists. In most cases, both are more interested in making sure their work is disseminated than in maximizing their return from it. In most cases, the target reader is a peer of the author. Publishing is designed to enhance reputation as well as to spread the word. Publishers must be careful to keep prices fair, lest they be seen as taking advantage of the goodwill of their authors, gouging the very customers who also produce their content.

In this kind of environment, you have to ask about the role of the publisher as middleman. No one who started as a self-published author and gradually developed all the infrastructure of publishing (as I did) can question the enormous added value that a publisher brings to the table. This value includes
editing (which starts with content filtering—the choice of what to publish and what to refuse—and extends through content development and quality control), manufacturing of the physical product, marketing, sales, distribution, and collecting and disbursing money.

In the early days of the World Wide Web, the rhetoric was that anyone could be a publisher. After all, with cheap, ubiquitous web servers, the cost of printing and inventory was minimized. There was a great deal of talk of “disintermediation.”

In a few short years, the reality has turned out otherwise. It is quite easy to put up a web page, not so easy to discover it. The fundamental job of publishing is precisely mediation: mediation between a huge class of potential authors, and an even larger class of potential readers. Simple mathematics dictates the rise of multi-tiered distribution chains, in which publishers aggregate authors, various types of resellers aggregate readers, and wholesalers aggregate publishers for resellers and resellers for publishers. The same multitiered distribution has emerged on the Web. Betting on this logic, my company created the first Web portal, a site called GNN (Global Network Navigator) in early 1993. We sold the site to AOL in 1995, and they later folded it into their main service, but the vision of web aggregators (i.e., publishers) has unfolded pretty much as I imagined it.

Many people with their own web pages end up writing for better-established web sites; those sites are further aggregated for readers by search engines, directories, and other portals such as Google, Yahoo! or AOL. In fact, web publishers now employ full-time workers to ensure that their pages are listed on these gateway sites, much as publishers of printed books employ sales people. A large proportion of Internet advertising has come from web sites trying to get better visibility for their product.
However, the Web does bring another wrinkle: the ability of groups to self-aggregate. The core functions of publishing, from content filtering to audience aggregation, can be performed by a group of interested users. This is particularly true when there is already a well-defined target community. This can be a disruptive force in the publishing marketplace. So, for example, sites such as Cnet and ZDnet spent tens of millions of dollars building and promoting portals for technical information on the Web, while two college students built a site called Slashdot (“News for Nerds. Stuff that matters.”) into a similarly powerful market presence simply by inviting their readers to submit, organize, and comment on their own content.

Interestingly enough, though, as Slashdot has grown in popularity and evolved into a real business, it has needed to add more editorial staff to filter the submissions of a growing marketplace of readers who now recognize that exposure via Slashdot is a powerful marketing tool. In short, even a community-centric effort ends up recreating some of the fundamental dynamics of publisher as middleman and aggregator.

What this evolution illustrates is that publishers will not go away, but that they cannot be complacent. Publishers must serve the values of both authors and readers. If they try to enforce an artificial scarcity, charge prices that are too high or otherwise violate the norms of their target community, they will encourage that community to self-organize, or new competitors will emerge who are better attuned to the values of the community.
April 2000

At the 2000 Waterside Publishing Conference, I was invited to speak on a panel called “Beyond the Book.” The conference is aimed at an audience of computer book authors, editors, and publishers. The organizers were no doubt expecting a few words on what O’Reilly Media is doing with eBooks, with the online sites we publish, like xml.com, and with our technical conferences. But instead, what I spoke about was why O’Reilly has always reached beyond the book in all of our publishing efforts. This article is a summary of what I tried to say.

Most publishers think that their business is to create products that people want, and that accordingly transfer dollars from consumers’ pockets to their own. At O’Reilly, we believe that the core of our business is to transfer knowledge from people who have it to people who need it. Yes, we are in business to make money, but this is a kind of housekeeping, not the purpose of the business.

I like to compare business (or life for that matter) to an extended road trip. Say you want to travel America by the back roads. You need gas for your car, food and water for your body. Especially before heading across Death Valley or the Utah salt flats, you’d better be darn sure that you have enough gas in your tank. But you certainly don’t think of your trip as a tour of gas stations! What’s the real purpose behind what you do?
Why then do so many companies think that they are just in the business of making money? At O'Reilly, our products aren't just books, conferences, and web sites: they are tools for conveying critical information to people who are changing the world. Our product is also the lives of the people who work for us, the customers who are changed as a result of interacting with us, and all the “downstream effects” of what we do.

When I started the company, my stated business goal was a simple one: “Interesting work for interesting people.” Above all, we wanted to be useful. Our financial goals were just to keep afloat while doing something worthwhile.

We started our business life as a technical writing consulting company. Between paying jobs, we thought we could perhaps create some value by documenting Unix programs that didn't have a manual. We wrote books like *Learning the vi Editor, sed & awk,* and *UNIX in a Nutshell* not because we thought there was a huge market waiting to be tapped, but because we used these programs, had learned about them, and thought we might as well put our skills to use passing on what we knew.

Our first print runs for the earliest of these books were only a few hundred copies. No other publisher touched these topics because they thought the market was too small. We didn't know enough to have such an opinion. All we knew was that people were using this software, and there wasn't much good documentation. And because we thought the subject matter was important, we kept promoting the books, rather than dropping them and going on to look for greener pastures. As it turned out, each of the books I mentioned went on to sell hundreds of thousands of copies.

Once we realized the possibilities for financial success in publishing, we tried not to forget our roots. Instead of pursuing the “hot” topics of the day, we kept trying to document software that we found interesting and useful. One of those things was the Internet, which was just emerging from academia into
wider use. Our book, *The Whole Internet User’s Guide & Catalog*, published in 1992, ended up having profound impact. It sold over a million copies, introduced many people to the Internet for the first time, and was a key factor in the explosive growth of the early commercial Internet.

One of the topics we covered in that book was the World Wide Web. We didn’t choose to cover the Web because our marketing department told us there was great demand for information about it! There wasn’t. At the time we published the book, there was a total of 200 web sites. But we knew the Web was an important technology and we wanted to spread the word. Once again, simply trying to do something worthwhile was the cornerstone of a great success.

It was at this time that we stumbled on an incredibly powerful approach to marketing. We had recently hired Brian Erwin, the former director of activism for the Sierra Club, to head our PR efforts. Brian helped us to realize that we shouldn’t be promoting our products. He showed us how much more powerful it was to talk about the technologies themselves, the possibilities that they might unleash, or the threats to the future that we might encounter. He reminded us that in marketing, as well as in product development, the best way to have an impact was to be useful.

We sent copies of *The Whole Internet User’s Guide* to every member of Congress, and launched a massive PR campaign about the Internet. We’ll never know how much impact our evangelization of the Net, and most particularly the World Wide Web, really had. After all, as they say, success has a thousand fathers. But we’ve heard, for example, that the NCSA team that developed the Mosaic web browser (predecessor to Netscape Navigator®) first learned of the Web from an O’Reilly direct mail piece. We also created the first web portal, a site called GNN (Global Network Navigator). GNN was the first advertising-supported web site (started in early 1993), and
was a prototype for sites such as Yahoo!, HotWired, and many other early web sites. While GNN vanished into oblivion after we sold it to AOL in 1995, its influence on the early Web was immeasurable.

In those early years of the Web, there weren't any venture capitalists looking for the big win. We invested millions of our own dollars in GNN, despite being a very small company at the time, and spent a lot of time trying to persuade big publishing companies that the Web was going to be important. Their narrow, bottom-line focus made them miss the biggest opportunity in any of their lifetimes. Those opportunities went to newcomers.

Our next big success was with what is now called open source software. We had published a book called *Programming Perl* in 1991. At the time, Perl was an obscure programming language with an enthusiastic following among Unix system administrators. As the Web took off, it became the dominant language for building dynamic web sites. Sales of the book exploded, and the second edition was one of the most successful computer books of 1996.

But something bothered me. Despite the evidence of its importance, Perl got virtually no notice in computer trade publications. There was no marketing budget, no PR agency, and so the press ignored Perl despite its grassroots success. We decided to spread the word. We organized a Perl Conference in 1997 not because we realized what a huge opportunity the technical conference business would turn out to be for us, but because we felt that the Perl language and the Perl community needed a voice, and a place to gather.

The success of the first Perl Conference got me thinking, and the more I thought about it, the more I realized that there were many technologies, like Perl, that were created as “free software,” but which didn’t get noticed by the mainstream. I knew
from the success of the books like *DNS and Bind, Apache: The Definitive Guide, sendmail*, and *Running Linux* that they covered important technologies. What’s more, these technologies had something unusual in common: they had all been developed not by commercial vendors, but by an idealistic network of independent developers.

I decided to organize a “summit meeting” of key free software developers, including Linus Torvalds (the Linux kernel), Larry Wall (Perl), Brian Behlendorf (a cofounder of the Apache group), Paul Vixie (BIND—the software behind the Internet’s domain name system), Eric Allman (*sendmail*, the Internet’s dominant email server), plus Eric Raymond, whose groundbreaking paper “The Cathedral & The Bazaar” had inspired Netscape to commit to releasing the source code for its next generation of web browsing software. The meeting also included people associated with the FreeBSD operating system and the Free Software Foundation’s GNU project.

It was at that summit that the group formally decided to endorse the term “open source software” instead of “free software,” which had negative connotations for corporate America. But most importantly, we were able to get across a story to the press, that much of the software behind the Internet, as well as the emergent Linux operating system, had come from an unrecognized wellspring of creativity, from people who were solving problems not for commercial gain, but because they loved what they were doing. These people wrote software to solve their own problems; they gave it away because they thought it might be useful to someone else. Their brilliance and their generosity was changing the world.

Most of you know what has followed. Within a few months, Linus Torvalds was on the cover of *Forbes*, with Larry Wall, Brian Behlendorf, and the Free Software Foundation’s Richard Stallman featured inside. Linux has taken off. And the idea, which seemed so novel at the time, that the Internet was built
on open source software, that programs like Apache, Perl, MySQL, BIND, and sendmail, are important far out of proportion to their nonexistent marketing budgets, is now widespread. And of course, sales of O’Reilly’s books on these topics have soared. Once again, we’d scored a marketing coup not by promoting our products, but by promoting big ideas.

My recent well-publicized tussle with Amazon.com over their 1-Click and Associates patents springs from the same roots. I worried that publically taking my biggest reseller to task might hurt my business, but it had become clear to me that the importance of the issue outweighed considerations of profit and loss. As it turns out, my efforts have strengthened the relationship between O’Reilly and Amazon.com, and Jeff Bezos has joined me in my campaign for reform of the software patent system.

In short, O’Reilly has become one of the pre-eminent computer book publishers not by “following the money,” but by following the trail of value. We publish books about topics that seem important, and they succeed because they fill a real need. We base our marketing and PR campaigns not on flash, but on substance.

A similar approach to thinking “beyond the book” is behind our other publishing ventures. I was once asked by an investment banker why my brother James and I had started Travelers’ Tales, when the travel book market is so much less lucrative than the computer book market. We did it because we saw a problem: tourism is becoming the world’s largest industry. It can be a soulless industry, one that treats the world as a theme park, with a list of attractions to be sampled, or it can be a thoughtful one that celebrates differences and rewards attention to the inner life of the traveler.

Travelers’ Tales has won over 50 awards, including (three times) best travel book of the year from the Society of American Travel Writers. Like many O’Reilly efforts, it’s also
spawned a host of imitation series from other publishers. It’s a thriving publishing company whose sales are growing at 20% a year. But that’s not what we’re about. We’re about changing the world, helping people get more out of their travels, and destinations more out of their visitors.

Why did we go into consumer health publishing? Because where is there greater “information pain” than when someone is hit with a life-threatening disease? Our Patient Centered Guides provide information from “health system hackers”—patient advocates who have experienced the best and worst of what the medical system has to offer, and who want to pass along their experience for sufferers of chronic or life-changing diseases. We’re not just in this business to make a dollar; we’re in it to make the world a better place. We look to make money so we can do more of what’s important.

But my point isn’t to brag about O’Reilly’s accomplishments in publishing or beyond it, though I am certainly proud of them. I am trying instead to urge each of you to think yourselves beyond the book. I want you to think why the technologies you cover are important, and how you can help to tell their story. Focus on making a difference, not on making a dollar, and I’ll lay odds you’ll make both.

Publishing is part of a noble tradition. It was born out of the same wellsprings as our great university system, out of the spirit of inquiry that brought us modern science. As authors, editors, and publishers, you are not just cogs in a money machine. You are scribes, capturing knowledge that might otherwise be lost; you are teachers, passing on knowledge that might otherwise go unheeded.

In short, I am here to tell you that what exists beyond the book is a world where you can make a difference.
Most issues aren’t black-and-white, though it’s often easier to fall into thinking of them in that way than to wrestle with subtleties and consequences. In this piece, written for one of our catalogs, I make the case that value is more than price, and that it’s in our self-interest to support what we value.

Like all technology revolutions, the Internet is profoundly disruptive. For all its promise, it can destroy old ways of life before we realize the value of what we’re losing. Part of the process of integrating new technology is developing social norms for its proper use. We have much to learn about how to use the Internet properly—how to avoid abuses such as spam, invasions of privacy, attempts by large corporations to control how we use technology, and the like. But there’s also a more humble area of disruption that I’d like to bring to your attention. Let me start with a story.

A few months ago, I was talking with one of my most loyal retail customers, a specialty computer bookstore in Massachusetts. “We survived the chains, and we survived Amazon,” he said, “but I don’t know if we’re going to survive the online discounters. People come in here all the time, browse through the books on display, and then tell me as they leave that they can get a better price online.”
Now, you might say, as the Hawaiian proverb notes, no one promised us tomorrow. Businesses, like individuals and species, must adapt or die. And if the Internet is bad for small, local retailers, it’s good for the online resellers and it’s good for customers, right?

But think a little more deeply, and you realize that my friend wasn’t complaining that people were buying books elsewhere. He was complaining that people were taking a service from him—browsing the books in his store—and then buying elsewhere. There’s a world of difference between those two statements. Online shopping is terrific: you can get detailed product information, recommendations from other customers, make a choice, and have the product delivered right to your door. But if you aren’t satisfied with the online shopping experience, you want to look at the physical product, for example browsing through a book in the store, you owe it to the retailer—and to yourself—to buy it there, rather than going home and saving a few dollars by ordering it online.

Think about it for a minute: the retailer pays rent, orders and stocks the product, pays salespeople. You take advantage of all those services, and then give your money to someone else who can give you a better price because they don’t incur the cost of those services you just used. Not only is this unfair; it’s shortsighted, because it will only be so long before that retailer closes his or her doors, and you can no longer make use of those services you enjoy.

If you like shopping in bookstores, remember this: many independent booksellers are on the ropes. (One store owner we know resorts to ordering books on personal credit cards when she is put on credit hold by publishers because she can’t pay her bills.) Even in the chains, computer book sections are in danger of shrinking in favor of other sections where sales are more robust. If you value the bookstore experience, my advice is this: buy where you shop. I buy lots of books online. I read
about them on a blog or a mailing list, and buy with one click. But when I shop for books in bookstores, I buy them there, and so should you. Don’t just look for the best price. Look for the best value. And if that value, for you, includes the ability to page through a book, support your local bookseller.

This story is the tip of an iceberg, of course. As with the unintended consequences of previous revolutions (pollution from automobiles and industrialization, for instance), it takes a strenuous forethought to make sure we don’t inadvertently damage parts of our world that we take for granted. It’s easy to get fired up about large technical, social, and political issues, but the future we create is even more the result of small decisions we make every day.
On Business and Life
In the late 80s, I wrote a draft of an employee handbook called Rules of Thumb that was never distributed. An employment lawyer I sent it to for review told me, “It’s the most inspiring employee handbook I’ve ever read, but I can’t let you use it.” I struggled with it for years, before giving in and letting the lawyers rewrite it. A watered down version was finally distributed to employees in the year 2000! This Introduction, from the earliest draft I could still find, dates from the summer of 1992.

What the Company Is All About

No one should ever be unemployed, and no one should ever be stuck with a job that they hate. There are so many worthwhile things to do, and it is possible for a creative, hard-working individual to make a living doing just about anything.

However, it is often hard to find a point of attack, and the risks of starting your own business are high. Many more opportunities are available to a larger organization with some financial strength than are available to an individual. What’s more, so many important things are the product of organized activity—people working together cooperatively to achieve something that no one could manage alone.

Ultimately, that’s one of the reasons why I founded O’Reilly, and what interests me most about it: it is a way for me, and those working with me, to have more choice about the things
we work on, and how and when we work, without living in a
garret. My goal is to build a platform that increasingly sup-
ports freedom for employees to demonstrate their initiative
and creativity, at the same time as it gives us the ability to tackle
larger and more interesting projects.

I don't think of O'Reilly as a publishing company. I see it rather
as a point of leverage between the skills and interests of its
employees, and opportunities for applying those skills out in
the world. Right now, because of our history, those opportuni-
ties are chiefly in publishing, but that need not always be the
case. I hope to branch out increasingly into other areas, led by
employee interests, and as we can make those areas work as self-
sustaining businesses. New ventures can start from either end:

1. Here's an opportunity. Who can take the ball and run
with it?

2. So and so is interested in thus and such. How can we
make a profit at it?

Season these alternatives with a dash of realism, and you have
my corporate vision.

As a new employee (or even one who has been around for a
while), it is very important to keep this in mind, because it
shapes company strategy and policy in many ways that are not
immediately obvious. Probably the most important is that
profit is not the most important company goal. Profitability is
desirable like good health is desirable in your personal life:
without it, nothing else goes well. But in any given case, max-
imum profitability might be sacrificed to the development of
employee potential, the pursuit of worthwhile goals, keeping
the company a place where people enjoy coming to work each
day, the freedom of employees to find their own balance
between work and personal life, or some other intangible. My
worst fear is that we would become a “professionally man-
aged” company with our eye only on the bottom line.
If I were to boil my company vision down to one sentence, it would be this: “To make money doing interesting and worthwhile things.”

Why “to make money”? Well, money is the fuel for the whole system. It makes the rest possible. What’s more, there is a wonderful rigor in free-market economics. When you have to prove the value of your ideas by persuading other people to pay for them, it cleans out an awful lot of wooly thinking. I think often of Alexander Pope’s comment about writing poetry in rhymed couplets. He said that funnelling his creativity through such a narrow aperture made it shoot out like water from a fountain.

Why “interesting and worthwhile”? Oddly enough, while dull people find very little to be of interest, intelligent people find almost anything interesting. (Curiosity is the wellspring of intelligence.) And so there are many things that can be interesting that do not add a great deal of value to the world. Adding the proviso that what we do be worthwhile in some larger sense limits our selections a bit, but like the need to make money, actually improves the chances of happiness at what we do.

A key point to remember is that the company is a means, not an end. The end is a better life for those of us who work, for our customers, and (as far as we can stretch it) for everyone we touch.
Fishing with Strawberries

A brief article I wrote for O’Reilly’s internal company newsletter in early 1995.

Recently, several of us were talking with an investment banker with whom we’ve been having some exploratory talks about finding strategic partners that want to figure out how to play this new Internet game.

The investment banker made a statement that really struck me, and the more I thought about it, the more I saw in it, both to agree and disagree with.

The statement was this: “You don’t fish with strawberries. Even if that’s what you like, fish like worms, so that’s what you use.”

He was referring specifically to finding out what the real needs of the potential strategic partners might be, since they might be focussing on something other than what we think is most important about what we have to offer.

That’s really good advice for any sales situation: understand the customer and his or her needs, and make sure that you’re answering those needs. No one could argue with such sound, commonsense advice.

At the same time, a small voice within me said with a mixture of dismay, wonder, and dawning delight: “But that’s just what we’ve always done: gone fishing with strawberries. We’ve
made a business by offering our customers what we ourselves want. And it’s worked!”

On one level, the difference between the two points of view is simply the difference between selling one on one to a very targeted prospect and selling to a mass market, where you are casting a wide net, and some set of potential customers will match your own “strawberry” profile.

But there’s perhaps a deeper level on which this difference is one on which a great deal that is special about this company hinges. We seek to find what is true in ourselves, and use it to resonate with whatever subject we explore, trusting that resonance to lead us to kindred spirits out in the world, and them to us.

I like to think that we have the capability to fish with worms when necessary, but that in general, we’re farmers, not fishermen, and strawberries go over just fine.
Some Necessary Qualities

I wrote this for our in-house newsletter in December, 1993, as I realized that we were starting to move from a company in which everyone knew each other and culture was based on personal relationships to one in which our values needed to be institutionalized.

As we approach the end of 1993, we’re possibly facing a greater rate of change than we did when we first came to publishing in 1985, or when I moved half the company to California in 1989.

With so much happening, we need all the best in ourselves that we’ve brought to the business so far, plus a lot of new strengths as well.

Creativity . . . Well, some people might say we perhaps have too much of that, and need to add a pinch more restraint. But in times when new projects seem to be bursting out of every corner, creativity may serve not just to start more things to overwhelm us, but instead to guide us in the next level of integration. The last thing we want to do is to rigidify in order to contain and manage the genies we’ve unleashed. We simply need to turn our creativity to new purposes.

Courage . . . We’ve broken a lot of comfortable molds in the last year. For example, we’ve broken out of the familiar X/Unix market where we’re well known and well thought of for a much wider market where we can no longer assume that customers know very much about us. This puts us back in a position where we have to really reach out to sell, and can’t
just sit back and wait for people to come to us. This can be uncomfortable—it’s a different level of cold calling—but is essential for a business that’s growing and not just resting on its laurels. Each of us who has customer contact now has to carry the whole message of the company and what it’s about, rather than just coasting on what everybody already knows about us.

Curiosity . . . To me, this has always been the foundation of the company’s strength, the quality that makes us look for interesting projects rather than just ones that we are already convinced will be successful. The desire to learn new things and solve new problems and then to share what we’ve learned has got to remain at the top of our list.

Kindness and caring . . . What do they have to do with business? Everything! Especially as the pace and the pressure pick up, we have to keep on trying to be good to each other, to our suppliers, and to our customers. What good does it do us to be successful by all the usual business indicators if what we are creating is unhappiness?

Foresight and good judgement . . . I think that we’ve always had a certain kind of strategic thinking that’s helped us to pick good markets and topics, and to be in the right place at the right time. However, we need to find out how to apply our vision to ourselves and not just to the outside world. What kind of organization do we need to have to grow into our next stage?

I could go on and list all the usual virtues, because after all, the same things that make the foundation of a good life make the foundation of a good business.

Still, there are specific challenges, and the biggest one is how to institutionalize the qualities that have up to now been carried for the organization by us as individuals. Especially as the pace of growth picks up, it’s increasingly easy for a company to be defined solely by its commercial goals.
It’s becoming clear that the company is going to keep growing, and to be around for many years to come. If that’s the case, one of our major challenges is to find a way to build company structures that support our underlying goals and values, to create a truly moral organization that is as effective as it is good, and will stay that way a hundred years from now, when all of us are gone.
Ten Years as a Publisher

A short piece I wrote for one of our catalogs in late 1996, reflecting on why we got into publishing in the first place. Our business has changed significantly since then, as has the marketplace, but there’s a thread of the perspective I express below that continues to run through the work we do.

O’Reilly & Associates began almost eighteen years ago now, as a technical writing consulting company, but it was ten years ago this past October that we made the shift to publishing our own books: 100 copies each of Learning the UNIX Operating System and Reading and Writing Termcap Entries stapled and in plain brown covers, went on sale at UNIX Expo in New York.

In celebrating our tenth anniversary as a publisher, I want to reflect on what brought us to publishing in the first place, what made us different from most other computer book publishers, and what I hope will continue to make us different through our next ten years, or twenty, or a hundred.

We had no grand plans for bestsellers, no sense even of where the technical book market was going or what the sales possibilities were. We were technical writers; our job was documenting things that needed explaining. In the cracks, when we weren’t busy writing manuals for hire, we decided to document some of the great free software that no one else was paying attention to.
The opportunity that drove us was not the opportunity to make our fortune (though certainly we hoped to keep the roof over our heads) but the opportunity to exercise our skills while filling a real need.

Over the years, we’ve tried to follow that same principle in selecting topics: if a book doesn’t fill a real need, let’s not spend time on it. This attitude tends to take us away from the mainstream, into areas other publishers are ignoring. So, when everyone else was doing DOS books, we did Unix and X. When Windows applications were hot, we did the Internet.

Now that the Internet is suddenly the central focus of just about every technical publisher, what are we going to do next?

To be completely honest, I don’t know the answer myself. The computer industry is in transition, with old paradigms and platforms crumbling and new ones jostling for the right to take their place. There are many clear needs, but also many candidates eager to fill them; things will have to stabilize before it’s clear what users are having persistent trouble with.

All of this is perhaps a roundabout explanation of why certain seemingly obvious books don’t appear in our catalog. Rather than rush out shallow, overlapping titles, we wait for the dust to settle, for the holes in the “obvious” to become apparent, and for a knowledge base to accumulate that is sufficient for us to publish books of enduring value.

In the meantime, we’re continuing to work on filling in the blanks in the undocumented areas we’ve already identified.

One consequence of writing books that meet real needs instead of following quick-buck fashions is that you stick with them. That’s why we’re still selling those first two books ten years later, when just about every other Unix book of their generation is long out of print. *Reading and Writing Termcap Entries* (now *Termcap and Terminfo*) is on its last legs, to be sure, as the technology it describes has largely become obso-
lete along with the alphanumeric terminals it supported, but *Learning the UNIX Operating System* has become a perennial bestseller, as have dozens of the books that followed it in the Nutshell Handbook series.

We’re going to continue doing books on Unix as long as there’s life in the “old geezer.” We’ve documented most of the old core programs, but new software is being written and coming into common use all the time. Even though the Internet is no longer solely a Unix phenomenon, it has sprung from the same freewheeling creativity and supports a rich ecology of free and commercial software—much of it poorly documented.

But when it really comes down to it, if you want to know where we’re going, don’t look for obvious buzzwords or books on the latest hot topic. Instead, look for unsolved problems, like how to make online publishing work as a business, or how to make web publishing (not just web browsing) a desktop tool as widely distributed as email. (Those are the problems that led us to launch GNN and WebSite, respectively.)

In short, we’re out there trying to gain the experience that will allow us to write the next generation of books you’ll be looking for: books about how to create and manage complex information products running on heterogeneous, distributed platforms.

I believe Joseph Campbell once said that the story of King Arthur and the Knights of the Round Table was the quintessential myth of Western civilization. The knights seek out not the high road to success but the deepest, darkest part of the forest, where there are enemies to fight and people in distress.

It may seem more than a little pretentious to compare writing technical books to slaying dragons, especially when there are still so many real dragons and ogres abroad in our society. But at the same time, we each play out myths like this where we are.
And the dark forest that we walked into ten years ago, largely by a series of chances, mischances, and small inspirations, was the forest of “information pain.” We don’t slay dragons. We write books. But we write books that are needed, and we take the time to do them well, so that we’ll still be selling them as long as they are needed.
Knowing When to Let Go

A piece I wrote for our internal company newsletter in November of 1992, in response to mounting stress at the company as we became more and more successful.

As I hear mounting tales of overwhelm—too much to do, too many things going right—from customer service, from marketing, from production, from editorial, (from my own psyche!), I am impelled to give a bit of contrarian advice.

When you feel yourself starting to clutch, to worry that you can't keep up, let go! There are times when the conscious mind can't keep up, but the unconscious will do just fine. The faltering steps of a newcomer to a dance become assured to the extent that she is able to forget the individual steps and yield to the music. The straining load of an engine getting up to speed gives way to a smooth hum as it slides into gear. I seem to remember (or did I make this up?) a wonderful martial arts book or movie in which an old master “bumbles” his way through a room, accidentally disposing of each of his highly trained opponents with strokes that might be luck but are more likely the highest level of skill guided by complete reliance on intuition. What I’m saying is meant in no way to diminish the huge increase in workload many of us are facing as a result of the company’s success. This is perhaps most striking in customer service, where they went from about 3800 orders in September to nearly 7200 in October—very close to a doubling of the normal workload. Such increases require urgent practical steps to keep things from going haywire.
Nonetheless, I’ll stick to my point: the conscious mind sometimes gets in the way, worrying about how to fit everything in. Periodically, I find myself about to explode, as I reach some kind of inner limit to how many things I can keep track of. If I’m lucky, a certain kind of letting go, a certain kind of surrender or forgetting what I know (and remembering the power of what I don’t know!) blasts it all into insignificance.

I find myself somehow deciding that I can’t keep up, and trusting my unconscious (or the hidden power of my spirit) to help me find my priorities. And often, I find that as soon as I do let go in this way, I find myself powering through the same tasks that before seemed insurmountable.

An old friend of mine used to refer to this as “jet” vs. “piston” functioning. A piston engine is overloaded at speeds that are hardly enough to engage a jet.

I also find that sometimes some of the things I was worrying about just aren’t that important. In that Rules of Thumb handbook I keep promising to finish, I quote a wonderful passage from Lao Tzu. He’s cataloging some of the qualities that are attributed to a wise man, and adds:

> And this one also: ‘Roiled as a torrent.’
> Why roiled as a torrent?
> Because when a man is in turmoil how shall he find peace
> Save by staying patient till the stream clears?
> How can a man’s life keep its course
> If he will not let it flow?

What I’m trying to say here is that we don’t need always to be in control, but rather, to be good at responding to change, to have the wisdom to know when to act, and when to wait “till the stream clears.”

I know that this article is already burdened with analogies, but let me close with one more. When bodysurfing (and in many other sports), you succeed by harmonizing yourself with forces
that are more powerful than you are, rather than by trying to force them to your will. That’s the kind of attitude I’m trying to urge on you now—not passivity, but rather an active surrender that finds the exact right moment and the center of the wave.
Walking the Kerry Way

My contribution to Travelers’ Tales Ireland, published in March, 2000.

I hadn’t spent much time with my brother Frank since he was about twelve years old, back in 1973. That was the year I’d gotten engaged to a non-Catholic, and my parents wouldn’t let me bring her home because “it would scandalize the children.” I was nineteen and equally sure of myself, so I refused to come home without her.

I finally gave in seven years later, when my father’s health was failing, and went home for a visit alone. After that, my parents also relented, and met my wife and three-year-old daughter for the first time. Our mutual stubbornness had cost us precious time together as a family, a loss made especially poignant by my father’s death six months later.

My relationship with my younger brother and sisters took years to recover. By the time I came home after my long exile, Frank was away at college, and thereafter we’d met mainly at family holidays and reunions. Still, we’d found many common interests and a mutual admiration. Both of us were entrepreneurs—I in publishing, he in construction—and both of us had struggled with how to build a business with a heart, a business that served its employees as well as its customers. In many ways, our lives were mirror images, seven years apart.
But there was one big crack in the mirror, one gulf between us that we skirted politely (most of the time): while I had long ago left the church, Frank remained a committed Catholic. He had also retained an abiding love for Ireland, to which he had returned again and again with my father, mother, and sisters in the years when I was persona non grata. He and my father had gone for many a tramp around Killarney, the town where my father was born, and where my aunt still lives. Mangerton, Torc, and the McGillicuddy Reeks were more than names to Frank; hikes on the slopes of these mountains were the source of the richest memories of his childhood and young adulthood.

I envied Frank the time he’d spent in Ireland with my father, and I’d always wanted to spend more time there myself. When my mother suggested that Frank and I might want to walk part of the Kerry Way together (a higher altitude walking version of the Ring of Kerry), we both jumped at the chance. I had a week between a talk I was due to give in Rome and another in London. It was March—not the best time to visit Ireland—but Frank could get free, and with his eighth child on the way, it was now or never.

We set out from Killarney on a blustery day. Though neither of us had done much recent hiking, we had an ambitious itinerary, about eighteen miles a day for the next five days. We were planning on staying each night at bed & breakfasts along the way, but we still carried packs with plenty of extra clothes.

The first day took us through Killarney National Park, up around the back of Torc, then down across the road to Moll’s Gap and into the Black Valley. The hike took more out of us than we expected, and we tottered the last few miles, grateful that our guest house was at the near end of “town” (a sprinkling of houses spread over the better part of a mile).

After a hearty dinner of local lamb chops, though, things began to look up, so when Frank confessed that it was his wife’s birthday, and that he wanted to go a mile up the road to
the valley’s only public phone, outside the youth hostel and
the church, to call her, I agreed to go along. It was pitch dark
by then, and raining to boot. We managed to stick to the road,
though, and eventually came to the phone. Unfortunately,
Angelique was not at home. How about going in to say a
rosary for her, he asked?

Now, I hadn’t said the rosary for over twenty years, and wasn’t
sure I even remembered how the “Hail Mary” went, but I agreed.

The church was open, of course, its outer door swinging in the
wind. In Ireland, at least in the back country, the church is
never closed. There was no electricity, and only a single candle
burning by the altar. The wind howled outside, the door
banged open and shut. We began to pray.

Frank helped me recall the words; the memories I’d never lost.
When we were small, the rosary, even more than dinner (where
my mother never sat down till everyone else had eaten), was
the time the family was all together. As we droned aloud
through the decades, the joyful, the glorious, and the sorrow-
ful mysteries, I remembered my father’s passing.

He had had a heart attack. He knew himself to be a dead man,
he said. He was met by Mary, St. Joseph, and surprisingly, the
devil. He begged for more time to make his peace with his fam-
ily, and his wish was granted. The doctors brought him back,
and as he lay in the hospital, intubated and unable to speak,
he was desperate to communicate with each of us, scrawling
on a small white slate. He wanted to reply to my letter, he said.

I had written him a few weeks before, telling him that even
though I had left the church, I had absorbed so much of him,
his belief, his moral values, his desire to be good, and to do
good. I didn’t want him to think he had failed. His short, so
poignant reply, written on a slate and soon erased, but burned
forever in my memory: “God forgive me, a sinner.” His apology
for the long years we had not spent together: “I only wanted
you to be with us in paradise.” The desire for togetherness in a world to come had become a wedge between us.

As he recovered over the next few days, he was a different man. He had always embodied for me so much of the stern, dogmatic side of Catholicism. Now, in the face of death, all that was stripped away, and the inner core of spirituality was revealed. His passion for his God was the heart of his life. How could I have never seen it before? So many of us build a shell around who we really are; our inner world is as untouchable as the heart of an oyster, till forces greater than we are pry us apart. Now, all was exposed. “I never showed you the face of Christ when you were small,” he told my brother James. Well, he showed it to us then. It’s as if he’d been turned inside out, and all the love and spiritual longing that had been hidden by his shyness and his formality were shining out like the sun.

Three weeks later, the time he had asked for was up. He had another attack, and this time he went for good.

We had taken him back to Ireland to bury him. It was a magical day; early April but beautiful as only a spring day in Ireland can be beautiful, a day of radiance stolen from the gloom. The funeral mass in the cathedral was concelebrated by thirty or forty priests: his two brothers, his childhood friends, and many others come to honor the life of one of Killarney’s dear sons now coming home for good. (He had himself studied for the priesthood before deciding to pursue family life instead; his brothers Frank and Seumas had become senior in two of Ireland’s great orders of priests, the Franciscans and the Columbans.)

He was buried in a Franciscan robe. He had long been a member of “the little order of Saint Francis,” a lay organization devoted to Franciscan ideals. We learned then of small penances he would do, like tying rough twine around his waist under his clothes. As if it were still the Middle Ages! I would
have scoffed, but I’d seen the light shining through him when impending death had pried all his coverings away.

Afterwards, the four sons, Sean, James, Frank, and I, walked behind the hearse up the main street of the town. As the funeral procession passed, those walking in the opposite direction turned and took “the three steps of mercy,” walking with the procession. The depths of Ireland’s Catholic legacy was never so clear as when a group of loutish youths, who might have been a street gang anywhere else, bowed their heads and turned to take the three steps with us.

As we turned up the road to Aghadoe cemetery, a breeze blew, and the blossoms fell from the trees onto the coffin. If it had been a movie, I would have laughed. It’s never that perfect! Except it was.

The cemetery, crowned with the ruins of a sixth century chapel, looks down on the lakes of Killarney. Ham-handed farmers (my father’s schoolmates) helped us carry the coffin over rough ground to the family plot. Normally, after the service, we would have all left, and “the lads” would have filled in the grave. But we wanted a last farewell, so we sent the lads on their way, and Sean, James, Frank, and I filled in the grave.

Now, twenty-five years later, I was back in Ireland. My tiredness fell away. I was at the heart of my father’s mystery, the place where he had turned his passionate heart to God, and the place where he had wrapped it round with rituals that had kept me from seeing its purity and its strength.

Somehow, Frank had seen through the ritual, had shared in it and sunk his roots to the same deep place. I was honored that he was opening the door for me as well. “Hail Mary, full of grace, the Lord is with thee . . .”

There are a thousand ways to God. Let us all honor the ways that others have found.
The next few days we wore our legs off, as the paths became wilder. The worst of it was the aptly named Lack Road, which our guidebook insisted had been used to drive cattle to market “within living memory.” We couldn’t see how you could drive a mountain goat herd across it now, as we picked our way down an impossibly steep slope. We understood why our aunt, who had worked in Kerry Mountain Rescue, had insisted we pack so many extra clothes. Turn an ankle out here, and you’re many hours from help, with changeable weather bringing freezing rain at any moment. At one point, the trail, which had us up to our knees in mud at many a point, vanished beneath ten feet of water, only to reappear tantalizingly on the other side, with no apparent way across. Ireland is a wilder country than many people realize.

On the fourth day, we came round the crest of a hill and saw the ocean spread out below us. Thirty or forty miles back the other way, we could see the gleaming lakes of Killarney, and amazingly enough, the green below Aghadoe. We could see many of the passes we’d picked our way through over the last few days, the miles that had lent soreness to our feet.

Along the way, we had talked through much of the old pain of the lost years, we’d shared dreams of the present and the future, but as we went on, we’d mostly fallen into a friendly silence. The old magic of Ireland was driving our reflections inward, recreating in us the unique Irish temper—passion and wildness and boggy depths alternating with conviviality, and ending up in quietness—a mirror of the landscape and the changing weather.
In The Meaning of Culture, John Cowper Powys makes the point that the difference between education and culture is that culture is the incorporation of music, art, literature, and philosophy not just into your library or your CV but into who you are. He talks too about the interplay of culture and life, the way that what we read can enrich what we experience, and what we experience can enrich what we read.

To make his point, I always like to cite an experience I had when I was fourteen, and had just read The Golden Warrior, Hope Muntz’s classic novel of Harold, last of the Saxon kings. Harold’s story fired my imagination, particularly the idea of the compact between a leader and his people, the compact that led Harold to march south to face William the Conqueror at Hastings, despite having just repelled an invasion by his half brother Tostig and his Viking allies up in Yorkshire. Advisors urged him to wait, but William was raping and pillaging, and Harold made a forced march to glorious defeat, keeping faith with his subjects. While reading this stirring book, I was vacationing in the Lake District of England. We swam every day at the swimming hole. But it wasn't just any swimming hole. This was the headwaters of the River Derwent, the river that ran red with blood when Harold defeated the Viking invasion. Book and place were together seared into my memory and sense of values,
each giving meaning and resonance to the other.

So there, I’ve told you about two books that made a huge impression on me. Sometimes, as with *The Meaning of Culture*, the book is a part of my regular mental toolbox. And with others, as with *The Golden Warrior*, there may only be a half remembered concept from decades ago. Here are a few of the books that have played a large role in my life:

*The Way of Life According to Lao Tzu* (Tao Te Ching), translated by Witter Bynner. My personal religious philosophy, stressing the rightness of what is, if only we can accept it. Most people who know me have heard me quote from this book. “Seeing as how nothing is outside the vast, wide-meshed net of heaven, who is there to say just how it is cast?”

*The Palm at the End of the Mind*, by Wallace Stevens. Stevens is my favorite poet, and this is the most commonly available collection of his poems. His meditations on the relationship of language and reality have entranced me for more than thirty years. I keep reading the same poems, and finding more and more in them. Also someone I quote often. Special favorites are “Sunday Morning,” “An Ordinary Evening in New Haven,” and “Esthetique du Mal.” From the last of these:

One might have thought of sight, but who could think of what it sees, for all the ill it sees? Speech found the ear, for all the evil sound, But the dark italics it could not propound. And out of what one sees and hears and out of what one feels, who could have thought to make so many selves, so many sensuous worlds, as if the air, the mid-day air, were swarming with the metaphysical changes that occur, merely in living as and where we live.

*Rasselas*, by Samuel Johnson. Johnson, author of the first major dictionary of the English language, is one of my heroes.
His work can be considered an extended meditation on Milton’s phrase from *Paradise Lost*: “The mind is its own place, and in itself can make a heaven of hell, a hell of heaven.” The quote from Johnson I subject people to most often is from his short novel *Rasselas*, in which a character remarks something like this: “I consider the pyramids to be a monument to the insufficiency of all human enjoyments. He who has built for use till use is supplied must begin to build for vanity.” The pyramids are actually quite a wonderful thing, but there’s a lot of wisdom in this analysis. Johnson’s work is a wonderful reminder that our minds have prodigious energy that must be focused on the right objects, and that much human pathology comes from having insufficient objects for our striving.

In that regard, I always like to quote from Rilke’s poem “The Man Watching,” which I encountered in Robert Bly’s collection of Rilke translations. The concluding passage, which talks about Jacob wrestling with the angel, losing, but coming away strengthened from the fight, goes something like this: “What we fight with is so small, and when we win, it makes us small. What we want is to be defeated, decisively, by successively greater things.” (I’ve gone back and checked, and that’s not an accurate rendition of Bly’s translation, but it is the way I retell the advice to myself and my friends.)

Colin Wilson’s *The Outsider* is another book that addresses the same theme: the untapped power of the mind and its constant battle with the world, to make sense of it, or be broken by it. But the book is also significant for me because at 23, reading this book, I wanted to write something as good as Wilson had done at that age. (For a wonderful story recapitulating Wilson’s ideas, I also recommend his takeoff on H.P. Lovecraft, *The Mind Parasites.* ) Wilson also shaped my relationship to books. So many critics write about literature and philosophy as a dead thing, an artifact. Wilson writes about it as a conversation with another mind about what is true.
An Introduction to Realistic Philosophy, by John Wild. The book that introduced me to the profundity of Aristotle. Virtue is the control of the appetites by right reason, the formation of good habits, or as my brother James once summarized it, “Virtue is knowing what you really want.” It was reading this book during high school that convinced me that philosophy was meant to be used, a guide to a better life, not a dry subject rehearsing the thoughts of dead men.

Science and Sanity, by Alfred Korzybski. Okay, General Semantics was the 30s equivalent of pop-psychology in the 70s, but there are some great concepts there. “The map is not the territory.” The idea is that people get stuck in concepts and don’t go back to observation. My friend George Simon applied General Semantics to psychology, and gave me a grounding in how to see people, and to acknowledge what I saw, that is the bedrock of my personal philosophy to this day. There are many popular introductions to General Semantics on the market, and also a fun science-fiction book, A.E. van Vogt’s The World of Null-A.

Rissa Kerguelen, by F.M. Busby. A science-fiction book I read at about the time I was starting my company, and that influenced me deeply. One key idea is the role of entrepreneurship as a “subversive force.” In a world dominated by large companies, it is the smaller companies that keep freedom alive, with economics at least one of the battlegrounds. This book gave me the courage to submerge myself in the details of a fundamentally trivial business (technical writing) and to let go of my earlier hopes of writing deep books that would change the world.

Islandia, by Austin Tappan Wright. A utopian novel written in the 1930s, about an imaginary country where technology has not yet hastened the pace of life, and where people find time to nurture relationships and the land they live on. Also a novel of “the long view.” My first Sun workstation was named isla, and the fantasy of living on the land was a part of my move to
Sebastopol. Physical labor is a wonderful antidote to the life of the mind.

*The Lymond Chronicles*, by Dorothy Dunnett. I discovered this series of six difficult, complex historical novels about a character roving the world at the turn of the seventeenth century as my company was passing the critical 50-person inflection point. Lymond is a brilliant leader who isn’t afraid of the opprobrium of his peers—he does the right thing, seeing further than those around him. He was a hero I aspired to emulate. The books are also just darn cool—the amount of historical scholarship packed into these stories is truly remarkable.

*Dune*, by Frank Herbert. When I got this book out of the library at age 12, my father remarked, “It’s sinful that so large a book should be devoted to science fiction.” Little was he to know that this book, full of wonderful concepts about how to come to grips with a world out of our control, would play so large a role in his son’s life. After I graduated from college, a friend who was editing a series of critical monographs about science fiction asked me if I’d like to write a book about Frank Herbert. I agreed, and it was that choice that set me on the path to becoming a writer. My first book, *Frank Herbert*, is online at [tim.oreilly.com/sci-fi/herbert](http://tim.oreilly.com/sci-fi/herbert). In the course of writing the book, I got far deeper into Herbert’s ideas than I had reading his books growing up. The core message of all Herbert’s work is that we can’t control the future, but we can control our response to it, surfing the edge of change and risk.

*The Structure of Scientific Revolutions*, by Thomas Kuhn. Kuhn introduced the term “paradigm shift” to describe the changeover from Ptolemaic to Copernican astronomy. But the book is far more than a classic in the history of science. It’s also a book that emphasizes how what we already believe shapes what we see, what we allow ourselves to think. I’ve always tried to separate seeing itself from the stories I tell myself about what I see. Pattern recognition is impeded if you are try-
ing to overlay an existing pattern on the facts rather than letting the facts sit quietly until they tell their own story. That’s General Semantics again.

As you can see, there are no books about technology or business on that core list! A lot of literature and philosophy instead. I apply myself to computers, social issues, and business with a toolset developed in another world. But that’s not to say that there are no books about science and technology that haven’t had a profound influence on me. Here are a few:

*Code and Other Laws of Cyberspace*, by Larry Lessig. One of my all-time favorite quotes is Edwin Schlossberg’s “The skill of writing is to create a context in which other people can think.” This book gave me a whole new set of tools for thinking about the complex interplay between four forces: government laws and regulations, social norms, technology, and markets. Lessig makes a simple but profound case that you can’t think of technical issues in isolation from their legal and cultural context.

*The Unix Programming Environment*, by Brian Kernighan and Rob Pike. In addition to its articulation of the Unix tools philosophy that is so dear to my heart, the writing is a model of clarity and elegance. As a technical writer, I aspired to be as transparent as Kernighan.

*On Writing Well*, by William Zinsser. I wouldn’t say this book influenced me, since my principles of writing were established long before I read it. However, it does capture many things that I believe about effective writing.

I don’t think I’ve ever read a business book cover to cover, but here are a few whose concepts have struck a chord or given me a vocabulary that helps me to see things in a new way or just to give context to my own ideas:

*The Innovator’s Dilemma*, by Clayton Christensen. An analysis of why great companies fail, because innovation often requires throwing out everything that has made you successful in the
past. Disruptive technologies are often born on the fringes, in markets where worse is better.

_Built to Last_, by James Collins and Jerry Porras. The idea here is that great companies aren’t afraid to have strong values. In fact, their cult-like values are what make them stand out from the norm.

_Positioning_, and _The 22 Immutable Laws of Marketing_, by Al Ries and Jack Trout. Anyone who wants to start a business with impact needs to read these books.

Building a personal culture out of what you read and see and hear doesn’t just end with books. I find moments in movies, in songs, and in pop culture that have also become part of my personal vocabulary for seeing and responding to the world. So, for example, in the mostly forgotten movie _Joe Versus the Volcano_, there is a scene in which Joe, played by Tom Hanks, is dying of thirst on a raft after a shipwreck. He sees the moon-rise, and says, “Oh my god, I forgot!” and has his faith in life renewed by the sight. So often, I see something special that returns me to myself, and I think of those words.