Backtraces

Three Decades of Computing, Communities, and Critiques

"The avant-garde are people who don't exactly know where they want to go, but are the first to get there."
—Romain Gary

It's commonly understood that computers, data, and the Internet did much to create the world of 2020 in which I am currently writing. These human contraptions resulted from engineering, financial, and ethical decisions made over many decades. Few people knew about these decisions at the time, and fewer remembered them later—but an understanding of the options available and the choices ultimately made can help us understand where we stand in our own time.

This is a memoir of 28 years spent in the computer field at the company known first as O'Reilly & Associates, then O'Reilly Media. I chronicled and often debated the decisions being made by programmers and their managers over those critical years, and I want the public to understand this history.

In many ways, our everyday lives reflect choices made in the computing field over the past few decades—even where we don't see computers in operation. Young workers crowd into a few overheated cities considered "innovation hubs", draining talent and income from the less lucky regions of the world. People rely on personal devices to get them everywhere and connect them to everybody and everything. None of this works without enormous companies gathering data that they push from place to place, and even these companies are not independent—they rely on nearly ubiquitous digital networks, which favor some geographic and demographic areas over others. Overworked creatives (often on contract) can order fresh-baked cookies from their dens at three in

the morning, but the same economic system takes food and housing away from millions. The state of public discourse disquiets everybody.

I'm not offering a grand theory for all this—that would be intolerable hubris, and incorrect. Rather, many small and often overlooked trends have wound tendrils around each other to culminate in our current situation. Some contradictions were resolved, others remain in raw contention. I want to lead you back in time from the moment where I am writing and let you stroke your hands across each issue, so that you come to appreciate what many people—brilliant, intrepid, devious, desperate, or aloof—did to create our world.

This memoir comes in a period when revisiting decisions in computing is not merely an exercise in curiosity. It is inextricably tied to our time of crisis, or rather of multiple crises swirling around one other, each in the desperate grip of the others. The questions posed by these crises revolve around technical issues, notably how to generate energy and how to intervene in biological processes. The topics in computing discussed in this memoir are equally salient. But although the technology in each question is central, the answers cannot be left just to the technologists. We all must gain some understanding of the issues and talk to each other about just and lasting solutions. I hope that this memoir guides people to deal more astutely with the problems of computer technology, law, and culture in years to come. If I manage to spin an entertaining story along the way, my satisfaction will be complete.

The usual disclaimers for a memoir apply here. The events I report are drawn from memory, and I am ready to admit that I made mistakes. I'm sure the people who participated in these events have their own viewpoints. Because I am offering a *memoir*, not a *history*, I take no responsibility for producing comprehensive or well-rounded accounts of events, but will fill in just the details that contribute to the lessons I want to convey. Still, accuracy is paramount to me. I have kept records of much of my activity, and consulted those records along with other sources wherever relevant.

Because I work in a technical field, you will not enter far into this memoir until you discover some tech talk about things such as cloud computing. Many readers may throw down the book at that point, thinking "This is not for me," or "This is going to be boring." But please persevere. Yes, this book is for you, and I hope to persuade you how much weight the technical issues carry for our times.

»»

An indented paragraph like this one indicates passages in the memoir that explain technical concepts underpinning the narrative.

☞ *Closing the book: A spiral of advancing strategy*

Table of Contents

Andy Oram
October 2020

Author's home page

If the computing history in this memoir interests you, check out some other articles where I cover specific historical topics:

From Unix to Linux: Key trends in the evolution of operating systems (November 13, 2020)
How the RESTful API became a gateway to the world (April 14, 2021)

The many meanings of Linux (September 17, 2021)
Open, simple, generative: Why the Web is the dominant Internet application (August 17, 2021)
Who's building businesses around free and open source software? (March 16, 2021)
How the Apache project boosted the free and open source software movements (March 24, 2021)
How free software contributes to cloud services—and what the cloud vendors give back (September 15, 2020)
Ten factors behind the popularity of microservices (July 8, 2021)
How the X Window System influenced modern computing (June 16, 2021)
Awk: The power and promise of a 40-year-old language (May 19, 2021)
How hashing and cryptography made the internet possible (September 20, 2022)
Fenced-off culture, the privatized Internet, and why book publishers lean on a 30-year-old doctrine (September 29, 2022)
The network neutrality debate: It all depends on what you fear (August 30, 2010)
Network neutrality: Distinctions and controversies (September 12, 2010)

Closing the book: A spiral of advancing strategy

Contents of this chapter

Distancing (2020)

"Were you surprised?"

Friends often asked this question, usually somewhere between the formulaic expressions of sympathy and the generic offers of assistance that are required in such situations. Their question was hard to answer. On the one hand, I didn't expect my twenty-eight-year tenure at O'Reilly Media to end as it did. On the other, signs of difficulties in our business environment had poked out their snouts during our most recent company meeting, three weeks before the layoff.

Every month or two, management invited employees up and down the hierarchy to an all-hands meeting. There the chief executive and financial officers, demonstrating the respect and trust that management had for their staff, shared sensitive financial data. They also used the all-hands meetings to rally us around the latest refinement of their strategy.

I took the subway down to join the audience in the Boston office. My wife Judy had traveled to Washington, DC a week before to share the sorrow of her family after the death of an elderly cousin. The next day, I would join her in a trip that we were afraid to return from a week later, because in a twinkling the COVID-19 virus erupted as a force from which we could no longer hide.

This all-hands meeting was in the first week of March, 2020. We all had been anxiously watching the news of COVID-19 challenging South Korea and forcing Italy to adopt strict isolation policies far out of line with its cultural norms. Analytical models

later suggested that COVID-19 was already rampant in Boston.

O'Reilly had announced the day before that it could not hold its California data conference, called Strata, which I believe was by then our largest event each year. Only two other computer conferences throughout the world had announced cancelations at that point, so our managers were almost unique in listening to the fears of our speakers and attendees.

Quick legerdemain by our conference team and technical staff rescued pieces of the Strata conference by moving them online. I was impressed once again, as I had been repeatedly over the years, by O'Reilly's resourcefulness and expertise. Not only could we offer some of Strata's strong content and training to the otherwise bereft attendees, but we were groping forward toward a model of online conferences that might anchor us in the future.

Here I can provide a bit of history, as I hope to do throughout this memoir, of events forgotten by most observers. O'Reilly had actually tried online conferences several years before, with disappointing results. Having attended dozens of face-to-face conferences and basked in their excitement, I understood why going online strained the medium. I had learned first-hand that the appeal of conferences lay in the magnetic energy that passed between attendees, ricocheting between the formal sessions and the gatherings in dining rooms and hallways.

Some of the most significant moments in my own career took place in the pregnant envelope of new encounters that clustered around conferences. I had shared my hotel room at a major health care conference with one of the leaders in the field, and bonded with him over his story of an amazing recovery from life as a drug pusher and jailbird. In the first hours of a conference on copyright, I met the person who later would give me the material for articles in two prestigious publications. Another conference on free software gave birth to a clever comic book aimed at drawing young people to computing.

Yes, face-to-face gatherings are the best. But the technology for online meetings has improved greatly, while our tolerance for expensive travel and for pouring airplane exhaust into our atmosphere has withered. March 2020 gave us a hint of a world where online conferences would be the only option for meeting.

Caution warned against complacency, though. There is no way for a company hawking online conferences to cajole the money out of attendees and sponsors that they would shell out for a face-to-face convention. The reduced costs of hosting the conference online could not rebalance the equation. Over the next two weeks, countries everywhere discovered the imperative of

physical distancing. The global conference circuit fell to the ground with a heavy thud, along with businesses ranging from cruise ships to jazz clubs.

Another part of this memoir lays out some of long-term impacts, not so obvious, of the closure of O'Reilly's conference business. The immediate effect is that, on the day when CEO Laura Baldwin announced publicly that the conference business was being disbanded, the company went through a large layoff that went far beyond the conference group. The ax fell in late March, scarcely giving one a chance to turn around after the all-hands meeting. The announcement also happened to come on the day when, on the Jewish calendar, I commemorated the death of my mother many years before. Thus, the yahrzeit of my mother will forever also be the yahrzeit of my career at O'Reilly.

I didn't immediately know who else had been part of the grand departure, but through the grapevine I finally learned about some of the notable positions eliminated. At hearing some of the names, I could not suppress a cry of amazement. These were people who drove the company forward—as I had done at times—through strategizing, data gathering, writing, editing, and more. Without them, the company might look the same, but it would take a different direction, operating under a new set of parameters.

Similar closures were going on around the world that month, a precipitous loss of jobs unlike anything seen before in history. I was one of 3.3 million people laid off that week. Most were suffering more than I, and it was impossible to focus on personal feelings during that time. Still, this I knew: The layoff deprived me of more than an income and a career in publishing. The O'Reilly that I had known was gone forever.

But what was this O'Reilly? How did it play such an outsized role in the history of computing? Indeed, how did the parallel and intertwining histories of O'Reilly, the computer industry, the Internet, and my own participation in these developments help to create the world we're in now? This memoir will attempt to cast light on these questions.

I actually got the idea of writing a memoir about my time at O'Reilly about four months before the layoff, in December 2019. Did I have some premonition of what was coming? I doubt it. I was feeling, in fact, more secure over the past year or so than I had felt for many years before. A string of highly successful and critically needed reports from my pen had recently demonstrated my continuing value.

But starting the memoir, which had reached about 7,700 words at the moment of the layoff, was a life-saver for me. Every time I sat down to add a memory to the log, I had to place myself outside O'Reilly, imagining that I no longer worked there. When

reality caught up with my speculations, I was psychologically ready to pick up life outside the role that defined nearly three decades of my life.

Programmers, when confronted with an unexpected result, enter a debugger and create a list of their program's behavior going backward from the end state: a *backtrace* allowing the programmer to examine the data at each point in the program's run. Similarly, this memoir will unpeel the historical layers in each major area of my career. The present chapter revolves around O'Reilly strategy, which itself evolved gradually in a spiral of unexpected turns.

The best brand to have (mid 2010s)

Strategy normally refers to a company's way to adapt its own activities with changes around it, but our company's strategy has prompted changes to the whole book publishing industry. During one period, the strategy led management to create an online service as a separate company called Safari. But our boldest forward-looking move in the 2000s was a backward one: to bring Safari in-house again. Shortly after that, managers decided the name Safari must go. The brand with which we'd subsequently talk about everything was O'Reilly, which has persisted through superficial name and logo changes during the company's whole history.

The choice of O'Reilly over Safari represents a continuing tribute to the founder who put his stamp on our culture and procedures. To emphasize the continuity, I minimize my use of official names (O'Reilly & Associates, O'Reilly Media) and just refer to the company as O'Reilly, a lowest common denominator that is shared in casual usage by employees and constituents alike. Although I'm not here to offer a biography of Tim O'Reilly, some of his traits that justifiably won him fame and followers will emerge through anecdotes that I scatter throughout the memoir.

The company O'Reilly occupies a unique position of trust in computer books during the feverish period of the 1990s when the Internet was stringing its cables through everything, and when the whole world—technologists, business leaders, and everyday computer users alike—was reading computer books to figure out where the hell we were being led by the new, uncanny intruders into their work and living spaces.

The name O'Reilly became the guarantee of quality as well as relevance. Once, out of curiosity, in the 1990s, my mother went into a bookstore and spoke to the clerk about whether they carried any O'Reilly books. The clerk answered, "O'Reilly is the

only publisher customers ever ask for by name." So it wasn't just like O'Reilly was the best of a class of publishers. It's more like O'Reilly was one class of publisher, and then there were all the rest.

Contrasts can tell you a lot about what people think. When some newcomer to the publishing industry did something notable, readers might recognize it by saying, "They're the new O'Reilly" or "They're doing the commendable work O'Reilly used to do." Many of these other companies released some good books and did fine in the industry, but there was never a new O'Reilly—except the new company that O'Reilly made itself into from one era to the next.

There was a deeper strategy behind the Safari name change, of course. Safari had marketed itself as an online content provider, but we needed it to be much more: a tool for staff development that would serve our business clients, something of a companion that would marshal their staff and provide education in all media toward the goal of deep business transformations. After the re-acquisition of Safari, O'Reilly could finally implement a plan called the integrated media strategy (IMS), first articulated several years earlier. The company could also wrench itself free from constraining atavisms in the publishing industry, which I will come to describe. Finally, we no longer suffered from confusion with the Apple web browser of the same name. Thus, it was a smart pivot to leave the name Safari behind.

With the re-acquisition, O'Reilly was entering a phase of radical transition that, given the volatility of today's markets, would probably never abate. The company's bold online strategy was meant to transform us in the minds of our customers, and therefore in our own day-to-day behavior, from a content provider to a Sherpa or guru who guides customers to their intended destinies. As the strategy came into focus, the company put new employee evaluation tools into place in the late 2010 decade. By aligning individual incentives with the goals of the company as a whole, we could benefit from each person being accountable. To bring the goals of every employee into line with our great vision, a new compensation system was installed, and my jousts with it produce an amusing story to end this section.

We're talking here of the last stage of my career, when my goals were reduced to raw productivity. I devolved into an editing or writing assembly line, fed raw material on one end and spitting out content on the other. Of course, my real responsibilities were reflected only dimly in page counts. I needed to grasp the direction taken by the technology I was writing about, the goals of O'Reilly as well as the sponsors who funded much of my work, and the best ways to reach the reader with complex, abstract concepts. I was an emissary of the company to everyone I encountered. My goals could in theory have reflected this subtle self-presentation, such as rating my ability to "Make sponsors eager to return and pay for more content." But nothing of this was reflected in my actual goals,

and I was very relieved to have it omitted. It's too hard to determine whether my actions helped or hindered soft goals, such as winning sponsors or even new customers.

The generic, overarching goals given to me, and that I was certain to fulfill every year—such as "Produce high-quality drafts based on input from authors or other parties"—should have relieved me from much interaction with O'Reilly's compensation software, but I could not stand outside the system. Quarterly measurements required me to deal four times a year with the same opaque, inconsistent, and mind-numbing interface that served others who were involved in planning and carrying out the company's strategy. And every quarter, something generated an urgent message to me from other staff. Often my attempts to fix the problems were ineffective or worse, and I resorted to asking human resources staff to tap into their sometimes esoteric expertise with the system and do the job for me.

Memorably, one January I was contacted by a distressed human resources person because my achievements for the previous year added up to 150 percent of my goals. This struck me as a resounding success—how could anyone complain? But the system wouldn't tolerate mathematically a success above 100 percent. The cause was obvious: Some goal had been duplicated and counted twice. But I had no idea how I had put the system into that state. Fixing it required a hidden web interface that the human resources team introduced me to. Eventually we got my achievements safely back to the same 100 percent that I had in every quarter.

The exhausting wheel: Continuous publishing (2010s)

The COVID-19 pandemic was not the first time O'Reilly was in danger. During 2001, the era of the September 11 attacks and the dot-com bust, we tumbled into a financial crisis and approached folding or being sold. That's when Tim O'Reilly brought in Laura Baldwin as chief financial officer. I don't know what discipline she exerted to turn the company around, but we recovered and rustled up some money to put into new projects. We entered a fertile period of experimentation, some successful and some not.

Huge efforts were invested in a process we called continuous publishing. We recognized that the most important projects, especially those fresh out of the egg, moved too fast for us to provide high-quality information in large books developed through standard, methodical procedures. Most of our books took longer than a year to write and release, whereas even an elapse of six months was too much. And we certainly weren't serving our readers to leave eighteen months or a couple years between editions. The need for change reflected several trends in computing.

»»

The squeeze on publishers caused by rapid updates was partly a symptom of the general speed-up in communications caused by the Internet, which furnished so much ground for O'Reilly content. Several related trends, in my estimation, spurred a sudden flowering of new software. Free software communities had learned to work together on large projects. Perhaps an even more important benefit of their collaboration was the prodigious outpouring of new programming libraries that reduced a lot of programming to plucking the proper function out of some free software project.

Programming libraries are simply collections of common-used programming techniques, usually organized around tasks such as math, web operations, Internet communications, or something else the library's developer programmer finds useful. The central importance of libraries can be seen by their role in choosing a programming language. Ask a bunch of programmers why they chose the language they're using on their current project. Few will point to some intrinsic quality of the language. Most will say that it offers a library they need. This reliance on libraries also lends a certain conservatism to programming, pulling back on the field's continuous lurch toward new languages.

Also, in my opinion, sleek new programming languages cut down programming effort, and improved test frameworks reduced the time required to ferret out bugs. I don't know whether research supports this conclusion. As we'll see later in the memoir, it's hard to find research in software engineering that generates trustworthy insights.

For a while, we dealt with the mismatch between reader needs and author capabilities by asking authors to write blog postings about new features, until time came to do a new edition. This workaround didn't solve the problem—the original book itself would quickly need a thorough update.

So we tried rethinking books significantly. No longer were they the fixed texts we had known through the millennia of books' existence. Instead, our books should be evolving repositories of the latest information on a topic. Our mission to institute this new viewpoint proved exhausting. It ran aground both human limitations and hallowed assumptions undergirding the publishing industry, which was in many ways stuck in the nineteenth century.

We were technically and organizationally ready for dynamic updates. (Our authors thought they were too, although we'll take a closer look at that momentarily.) It was other factors pervading the publishing industry that held us back. The degree to which

the publishing industry is stuck in centuries-old thinking was shown by their repeated talk of replacing "brick-and-mortar stores" with online offerings. Very few bookstores are housed in brick and mortar anymore; their buildings are made of concrete and steel.

Part of the new initiative was successful: We could offer customers a series of pre-released online books, which we called early releases or rough cuts, that included whatever chapters were finished at the moment. Each chapter might or might not have undergone editing and tech review before inclusion. Readers appreciated access to this early material, and it proved a powerful selling point for each book. We even accepted comments from readers, but I never saw a useful tech review comment come from an early release. At best, we got reports of trivial typographical errors.

The post-release phase was where our strategy failed. Let me give just a couple examples of the publishing hurdles that led to intense discussions, as we strove toward solutions that didn't materialize.

»»

First, take the ISBN. That's the publishing industry's equivalent of a universal product code (UPC) or European article number (EAN). Most readers don't associate books with ISBNs—for instance, they don't usually make it into book reviews. But distributors and retailers depend on the ISBN. It is the treasured seal borne by a book from conception through final sale. A hardcopy book will have one ISBN, the paperback of the same book another ISBN. Each new edition gets a new ISBN. There is no linking or continuity; there is only distinction without nuance.

The rigidity of the ISBN has long been a marketing problem for online sales. For instance, we found that we could build up hundreds of positive reviews on Amazon.com for a book, and then lose them and have to start from scratch when we put out a new edition. Amazon.com by then had established its dominance as the world's major retailer of books, and was branching out into other areas of commerce. For a while, people searching Amazon.com for a book would pull up an old edition that was highly rated, and fail to see the new edition because it had just started to build up ratings. In a field where buyers prioritize the recent date of information as a key to relevance, that design flaw could kill sales.

As another example of the resistance to our continuous publishing strategy that's inherent in the book industry, Amazon.com penalized publishers for putting out new editions too often. One of my authors based an excellent and very popular book on some free web-building software, which made it easy for readers to get started with a few clicks. It was his rotten luck that the software was taken down by its distributor right after the book was released. (A

problem that could come up only because the software was proprietary: free for download, but not truly free and open source. I'll explain this further in a later chapter.)

The author quickly found a new freely downloadable package and rewrote the book around it, but the algorithms at Amazon.com flagged the new edition as suspicious. The British subsidiary (alas, the author happened to be in Britain) took the book off its site altogether. In this and other perplexing situations, our marketing team was left with the unpleasant task of explaining arcane retail behavior to irate authors and editors.

In such an inhospitable terrain, we couldn't assign new ISBNs to updates. Supposedly, we had a work-around: Because we could issue books on a print-on-demand basis, so that few or no old copies would lie around the warehouses of Amazon or another distributor, we could simply ship the new version without saying anything. But then, of course, the marketing on the retailer's site couldn't reflect the changes, which would concern potential readers a great deal because it would influence their decision to make a purchase. The retailer couldn't even indicate to readers that the material was current.

I can't wrap up my litany of complaints about publishing without a reference to the dreaded ONIX classification system. ONIX is a standard set of keywords used by publishers and retailers throughout the industry to categorize books. Bookstores, including Amazon, relied on these keywords to fit books into their recommendations and search results. So my marketing team repeatedly stressed the critical task of choosing the right keywords. The problem is that authors and editors didn't get to choose keywords. We were prisoners of a system that appeared to be some 30 years out of date. Computing subjects that were nearly forgotten decades ago still appeared on the approved list of words, which totally omitted the hot topics on which we were urgently pushing out books. So every time a book came up for release, the requirement to choose ONIX terms made me want to scream.

The mephistophelian ONIX system sharpened my interest in knowledge bases, ontologies, and taxonomies, which a rapidly growing Web was trying to build in various ways. I don't need to bore you with a list of technologies that have been used to relate pieces of information to one another—to create a rational structure for a world bursting with disorderly innovation—because modern ontologies allow you to find them yourself through web searches. Ontologies are the modern version of Hermann Hesse's Glass Bead Game, the prophetic technological invention in his final novel. (Hesse, true to his philosophy, rejected the promise of a universal data processing system.) Like other models, no ontology is perfect.

I have to relate another story to explain my cynicism about ontologies. A friend of mine once proposed a book on how to create them. Finding herself going around in circles as she tried to capture her process for developing an ontology, she brought on two other industry leaders. That didn't help: They just collectively continued to go around in circles. I commented in detail on their drafts, as I did with every author, trying to extract from haystacks of confused text some needles of truth. The authors finally canceled the book, admitting that they didn't understand their own processes and couldn't arrange those processes into a set of coherent guidelines.

Back to O'Reilly's headaches with the publishing industry. How did we finally resolve the issues? We ultimately left traditional publishing (while still allowing our books to sell through retail channels, as a sort of by-product of our content generation) and focused on the online platform, where we had total control and where distribution issues around ONIX and ISBNs were moot.

But continuous publishing wasn't destined to be a lasting strategy anyway, because of the other consideration I mentioned: human limitations.

When we were trying out continuous publishing, I prepared my authors for it well in advance of completing their books. They generally liked the idea and wanted to keep their books up to date. And why not? They must have loved their topics—otherwise they wouldn't write about them in the first place. Of course they would keep following developments in their field. And why wouldn't they want to put in effort to keep their books relevant and selling?

All these commendable intentions fell apart by the time they finished the book. A full-length book is an unfathomably gargantuan undertaking. I had often sensed in my own body the burn-out authors felt toward the end. They needed to recover from the effort, to get back to family, work, and professional commitments. So they just had no mental cycles for keeping a book up to date. Basically, we were asking them to continue the extraordinary, draining process of finishing a book forever.

Occasionally, an author would turn in one desultory update. I think one or two authors actually did try to update their books faithfully, but these books weren't selling well and the exertion just wasn't worthwhile. Other editors must have discovered the same problems, because continuous publishing quietly disappeared from company talk after months of intense discussion and preparation.

Continuous publishing, although we couldn't pull it off, showed our cleverness and agility in responding to a changing market. It was made possible by our unique platform for automatically generating electronic books in multiple formats, and that platform had a crucial role to play later. We return to it in the next section.

Our next innovation to deal with the publishing landscape of the 2000s was Make Magazine, which was thought up and launched in 2005 by Dale Dougherty, second in command to Tim O'Reilly. I believe Make to be one of the most significant responses by our company to issues of the day—more significant perhaps than our championing of open source, because we had a big impact in a relatively young field.

Dougherty was picking up on a new Do It Yourself (DIY) attitude that was entering a number of fields. He focused on grassroots electronic and computing experimentation such as aerial cameras (with some departures into purely mechanical products). They quickly took up the new quick-and-dirty embedded devices enabled by cheap, mass-produced computer boards. Our company separately tracked other grassroots, DIY movements such as biochemistry. I myself visited the BioCurious lab in the Silicon Valley, one of the most vibrant outposts of the DIY biohacking movement, and interviewed one of its founders, Eri Gentry.

»»

About the same time Make Magazine started in 2005, an exciting new embedded system for artists, hobbyists, and engineers was released by a university team in Italy. Called the Arduino, it was a processor board that could run on its own and also slip conveniently into mechanical or electrical contraptions. The Arduino could do such things as water houseplants on schedule or watch over a gate and signal the owner when a potential intruder arrived. The Arduino was completely open source hardware. Another popular system called Raspberry Pi came later; it ran Linux and its specs were publicly available. A wealth of free software ran on both. Cheaper and easier to set up than traditional embedded systems, these boards were ideal for educational purposes, experiments, home projects, and prototypes for entrepreneurs with ambitious plans. Arduino and Raspberry Pi exemplified the DIY philosophy and brought it to any activity that could be digitized.

DIY electronics and computing offer huge potential for human development. The research of Eric von Hippel—highly relevant for free software—revealed that many important products were created by the users of commercial offerings, and were incorporated into the new official products by manufacturers who watched what their customers had created. In other words, innovation that we ascribe to manufacturers often comes actually from their customers.

Tim O'Reilly also had a favorite analogy that came up in his talks over the years. He'd say that a lot of DIY energy early in the twentieth century, up to maybe the 1970s, went into modifying ("modding") one's car. The same ingenuity was applied by a subsequent generation to computers.

Dougherty and his team identified, celebrated, and pushed forward this social movement with Make Magazine. They started up a garage-style lab in the Sebastopol, California campus at O'Reilly. To me, visiting the Maker space felt like leaving the dull grouped desks of a conventional office (the Sebastopol managers had adopted the trendy open seating style) and enter a mad scientist's fantasy lair.

Make Magazine was an extremely audacious venture, attempting to succeed at print publishing in an era when scads of print magazines were moving online or shutting down. This Maker division was probably the most dynamic part of the company during those years. And as Dougherty reported at one company meeting, it wasn't fair to work for Make Magazine because they had a totally disproportionate share of the fun.

The Maker department also started the Maker Faires, where people could demonstrate truly magical creations. The concept was adopted throughout the world. No verbal description could bring to life the experience of being personally present or these outbursts of talent and technical dexterity. Digital cleverness mingled with wild mechanical concoctions, such as cycles driven by ten people and fire-breathing metal sculptures that recalled the famous Burning Man desert festival.

That's all I have to say about two of O'Reilly's innovations in this difficult period. The third was the integrated media strategy (IMS), which expanded us from books to conferences and then to all kinds of media, such as video and interactive online learning. As mentioned earlier, this really came to fruition with our repurchase of the Safari company and integration of content production with delivery.

Electronic publishing and trailblazing with integrated media (2000s)

A large geographic split between offices was pretty rare in small companies when Tim O'Reilly moved from the Boston area to Northern California around 1992 to break ground on his Sebastopol office. Before his move, staff worked in an office building in Newton, Massachusetts, and, at the very start, a barn on Tim O'Reilly's Newton property. The barn remained a legendary origin story that I missed out on.

When Tim relocated his family to California, he took most of his staff with him, but opened an office in Cambridge for those who wanted to stay in Massachusetts. Those who sought their fortune in the wild West (Sebastopol was still known for apple-picking more than high-tech, but now everybody supposedly works in computers or renewable energy) and those who chose to remain cleaved into rather natural divisions of labor. The productive forge of the O'Reilly & Associates company—editors, production team, the artist, the print coordinator—occupied the Cambridge office. Marketing, customer support, and other necessary but secondary functions relocated to California. Occasionally staff would grumble about a clash of cultures, but I felt untouched by that, perhaps because I found ways to spend a lot of time in the Sebastopol office. By visiting regularly I could renew my friendships and feel that I was participating in O'Reilly as a whole, while other staff were stuck in one office or the other.

Because the split took place in such a young, burgeoning organization, entire departments would grow up around the individual staff people who chose to stay or go, and I think the seemingly arbitrary division based on personal geographical preferences ended up benefitting the company. The West Coast office recruited people who were attracted to a life out among the trees and the neighborly California counterculture. (I imagine a lot of this lifestyle changed as population, traffic, and housing prices increased.) The East Coast office appealed to creative types who craved the bustle and heavyweight intellectual arenas of Boston and Cambridge.

Among the benefits of our bifurcated location were the long-distance working relationships we learned to cultivate, which prepared us for a new world of globalized creativity. And our books contributed to building that world. Being forced to use the primitive collaboration tools of the 1990s, such as the FTP file transfer protocol and email with minimal graphics, along with the need to adapt to different time zones, gave us the expertise to deal with authors, reviewers, and technical experts anywhere they lived. We were very alert to the power of the Internet, an awareness that paid off in our first real blockbuster, *The Whole Internet*, which Mike Loukides edited and drove to fame in 1992.

The critical role of the Cambridge office in writing (when done in-house), editing, artwork, production, and all essential aspects of getting our product out did not immunize us against decisions made by individual managers with personal agendas in Sebastopol. At least twice, in the design group and the tools group, a West Coast manager felt incapable of dealing with staff outside their office and insisted on consolidation. Some of the Cambridge office staff were summarily laid off, while others were given the option to move West but—as they told me—without reimbursement for their relocation expenses.

Not a single staff person in Cambridge who was placed under the tool manager or design manager in Sebastopol lasted through the consolidation. I don't believe these managers took the pulse of the company or explored the ramifications of flexing their managerial biceps. They probably had no inkling that they were discharging some of our most creative and accomplished professionals. Geographically distributed departments were unusual at that time, and organizations lacked the understanding as well as the technical tools we have today to manage remote work. But I'm still puzzled that upper management failed to rein in the managers and veto their activities.

Both functions under attack, design and tools, were intimately entwined with the other creative activities for which the Cambridge office was responsible. Naturally, both functions crept back to that office after the original staff were deposed.

Tools provide a particular educational example of the ways creative teams fill in skills gaps. This story provides an even stronger lesson about the opportunities that a fluid company like O'Reilly can provide to someone with vision.

The tools group reconstituted in Sebastopol proved incapable of solving everyday problems that turned up routinely in Cambridge. I don't ascribe this failure to geographic distance so much as a lack of listening—a lack of regard for the concerns being expressed in the Cambridge office. After a while, back in Cambridge, a junior production person named Andrew Savikas—watching requests go out toward the West Coast team and perish somewhere in between the prairies and the Rocky Mountains—started to collect tasks from his fellow production staff and write small scripts that solved their problems. Without recognition at first, Savikas single-handedly recreated a tools capability in Cambridge.

As his purview grew and upper management started to give him responsibility and resources, Savikas looked at trends in the publishing industry and realized that we needed a whole new writing environment to meet our business goals. Essentially, he gave us the tools to write books in a standard format called DocBook XML. And here a historical digression is in order.

O'Reilly was a hardy native plant in the stony but nurturing soil of Unix, an operating system that represented the boundaries of the entire known universe to most hackers and researchers. (Mention Microsoft Windows to these people and they'll say—well, just take my advice and don't mention Microsoft Windows to them.) Documentation in that world used a format called troff, pronounced Tee-Roff, that has probably never been surpassed for rigidity and inscrutability. Because documentation for the X Window System in the 1980s used one of the Unix troff

formats, we could quickly adapt their manuals for our own publication. So there were modest benefits to sticking with it.

The basic troff syntax was a period at the start of a line, followed by one or two characters. The two-character restriction saved precious memory and disk space at an early stage of computing, as did the fearsome Sendmail configuration files with which every Unix system administrator wrestled. Although one had to limit any new troff commands (known as macros) to two characters, these characters did not have to be letters or even digits. Thus, inventive troff designers used all manner of weird punctuation in addition to inscrutable sets of letters and numbers, In general, troff code was as readable as a Sendmail configuration file, a notorious jumble that few humans could even start to parse.

O'Reilly clearly couldn't impose troff on all our authors, so we also produced books in FrameMaker, a doggedly proprietary product of 1980s vintage. We could accommodate nearly all authors by giving them either a license to FrameMaker or a Microsoft Word template. Microsoft Word was so prevalent that one could count on everyone having the software, or some other software that could produce the same format. One of our tools staff with a fondness for TEX (software created by programming legend Donald Knuth and widely used in academia) created a specific O'Reilly template for it.

But during the 1990s, managers at O'Reilly realized that online books would soon become a central part of our offerings. We even prepared for the possibility that all content would go online. The popular Kindle and ePub book formats were yet to be invented, but we were girding our loins for an online disruption of publishing. There were two reasons for our huge investment in a transition. First, the availability of free online content—and most importantly, the Java documentation from Sun Microsystems, given the dominance of the Java programming language at that time—raised formidable competition. Second, we expected that the public would get accustomed to reading online and would want books delivered that way. The advantages of the online medium for reaching international markets cheaply and instantly were also enticing.

Dale Dougherty, whose prophetic insight in regard to Make Magazine was described earlier, led an initiative in the 1990s with twin technical and corporate goals: firstly to develop a standard that was equally good for print and online publication, and secondly to sign up other far-thinking people in the publishing industry to adopt it. We had to keep changing this standard, though, to keep up with the best of publishing technologies.

»»

The handiest format used by large companies in the late twentieth century for documentation was Standard Generalized Markup Language (SGML). Even Tim Berners-Lee turned to SGML for guidance when he designed his HTML format for web pages. Basic SGML conventions survive in HTML, such as angle-brackets around tags and ampersands to start special characters.

After a decade or two of use, the burdens of SGML were apparent. It was overly complex and hard to write programming tools for. As one example, SGML boasted of being highly structured, nesting elements (such as paragraphs within lists) in a very strict way, but then went ahead and undermined this discipline by allowing the end tags to be omitted under some circumstances. HTML carries over the less disciplined aspects of SGML and violates the conventional rules in many other ways.

Eventually, web developers created a format called XML that was easier to support than SGML. They encouraged web designers to use a structured form of HTML called XHTML. But most web developers by that time were using WYSIWYG tools to create their web pages. The tools had been designed with no appreciation for structure or discipline, and had wandered too far from robust coding practice to be upgraded so they could take advantage of HTML's or XHTML's structure. A look at how web page designs are rendered into HTML by popular tools is like watching an eighteenth-century typesetter count out spaces and dashes. Doughery's group designed a template they called DocBook around SGML originally, switching to XML by the time they were ready to release it.

XML itself proved to be more heavy-weight than most users needed, while version 5 of HTML provided stunningly powerful web features such as canvases for drawing. So a number of years after investing so much in the creation and promotion of DocBook, we relegated it to legacy books and started using a lightweight format that made writing easy in a plain text editor, along with a simple Web interface like that offered by blogging sites.

We had audacious goals, such as distinguishing different types of words so that we could refine searches. DocBook had not only a tag for emphasis, but one for file names, another for commands, another for URLs, and so forth. These were all rendered in italic font, but we initially tried to mark every word with the precise tag so that we could exploit the distinctions in sophisticated tools that we meant to develop later. Our tools team even learned how to process italic from Word files using tools that could guess quite well when an italicized word was a file name, a command, or a URL. But nothing ever came of all this arcane differentiation.

Format changes aside, our platform was robust and powerful, able to spit out high-quality content in all the popular electronic book formats, including PDFs for printing. When Amazon.com's popular Kindle ebook came out in 2007, we quickly accommodated its format. This was crucial, because Amazon.com was the first major company to enter the digital book market, and they had the overwhelming market clout to make their format a must-have. Other publishers were trying to provide books for e-readers through miserable accommodations such as scanning pages and using optical character recognition (OCR), which could easily be detected by its ugly formats and the computerized errors it left.

Savikas, the young production visionary who revived our tools group, was recognized by upper management and put in charge of a team to write new tools for manipulating our DocBook standard and rendering it into the formats we wanted. In addition to creating a smooth path from the tools used by authors to the deliverable sent to the reader, his team grounded our workflow on the reliable processes that programmers used to ensure accurate code.

Savikas went on to many more achievements. In order to cull ideas from around the publishing industry and pursue new opportunities given by the Internet, he launched a conference called Tools of Change, integrated into O'Reilly's conference offerings. As the acronym TOC suggests, the conference was meant as a "table of contents" for those interested in transforming the media industries.

Our company was so proud of what we were doing with TOC that the entire Cambridge office was invited to one of the conferences in New York City, all expenses paid. The experiments on display there were quite impressive. Most involved multimedia. But of course, it takes a lot of money and specialized expertise to create large amounts of graphics, video, or interactive content, even leaving aside the challenge of integrating them in ways never done before and creating an experience that delights or educates the viewer. The fine works shown at TOC remained mostly one-off demonstration projects. The publishing industry has not made a wholesale turn toward integrated multimedia—although many video games show impressive sophistication, and some online interactive teaching systems have used new media well.

Savikas ultimately joined the Safari board and then became head of that company. His career is a kind of Horatio Alger story appropriate for Silicon Valley style innovation, and in my opinion showed O'Reilly's staffing policies at their best.

The results of our corporate transformation were pretty spectacular. We managed finally to create a "learning platform" that guided the perplexed through the steps and tools they needed to pick up the confusing new technologies that were required for

success in our software-dominated world. I believe the term "learning" could apply not only to the customers using our platform, but the platform itself: Data collected on use could improve its recommendations. Later, we integrated other media into our offerings, but I rarely participated in that. Sponsored content became part of our integrated media strategy, and that will be the topic of the next chapter.

Even the completed merger between O'Reilly and Safari took a long time to carry out our integrated media strategy. And along the way, the concept took a leap in sophistication. A year or two into the merger, Laura Baldwin started talking about how it wasn't enough to offer great content, or even great content in a variety of media. She called the current Safari a "library" and said we had to be more forceful about helping our clients take advantage of our learning opportunities.

Gradually, during all-hands meetings that took place every month or two, a complex strategy was unveiled. We would prompt people on "learning paths" to proceed from one book or video to another. The learning paths could be as simple as a list made up by a reader who felt that their sequence of learning could help others. Over time, learning paths were supposed to be tailored to individuals by tracking and analyzing their behavior.

I don't know whether this AI-rich strategy took off. Clearly, it was inspired by the recommendation systems used by online media giants of the day—Amazon.com, Netflix, YouTube, and so on—although I don't believe O'Reilly management mentioned that. Recommendation systems, obscure and highly prized by their designers, lurk inside everything Internet-related as I write this: retail, social media, search. The recommendation systems make all those things work, while threatening to wipe out independent culture and productive discussion by pushing all consumption toward a few fortunate choices that quickly get outsized. Recommendation systems enforce the evangelist Matthew's words, as expressed by Billie Holiday, "Them that's got shall get, them that's not shall lose."

The O'Reilly vision was a hundred times cooler and more brilliant than the strategies of the retail and streaming media companies. What the others promoted was a set of lateral moves: You watched one thriller, so they showed you another, and another. You weren't improving, you were just passing the time in hopefully enjoyable diversions. But O'Reilly's strategy represented actual progress: As you moved from one tutorial to another you were approaching a goal.

Although I was insulated from the planning around our IMS, I think it helped me a great deal personally. The IMS encouraged sponsors to think about a hierarchy of materials ranging from blog postings to conference sponsorships. When they

sponsored written content, they created new work opportunities for me as either writer or editor.

The new strategy committed us to focusing on business-to-business (B2B) instead of the old business-to-consumer (B2C) model of selling individual books. It unveiled an awe-inspiring concept: helping a business decide what its technical staff needed to learn, and guiding entire departments through their education. Dashboards and department-level tracking were to be involved.

Scarcely a library.

Experimentation and expansions of series (mid 1990s)

The 1990s saw its own set of threats to publishing—as well as journalism and other existing media—that demanded quick and bold responses from practitioners who wished to survive. And survive we did, thanks to a mature culture of experimentation. Let me list here some of its forays into new topics, formats, and approaches to human psychology.

My first example is Jeffrey Friedl, who briefly became a superstar after writing a unique book about regular expressions. Originally a small corner of the tools for text processing, this method of generalizing searches in text became a part of the average programmer's skill set after being built into the Perl language, and then copied by other scripting languages. You don't have to understand regular expressions to appreciate the story of the book, but I'll explain why there was so much buzz about them.

⋙

Any text editing tool lets you search for a particular word or phrase. Regular expressions were a little mini-language that made this kind of search almost infinitely open to generalization. You could describe a form of the text you were looking for, such as "three digits followed by a hyphen and four more digits" (a common way to represent telephone numbers in the U.S.).

Think a minute about the millions of bytes flowing over the Internet second by second, and how developers want to find content of interest in these rivers of data. Regular expressions become a necessity.

The term "regular expressions" itself has an arcane mathematical origin that was irrelevant even to the original Unix implementation, and many prefer the term "patterns". In their various forms, regular expressions came to be indispensable as the Web cultivated increasing amounts of digital text.

Friedl had the most creative approach to layout of any author I ever worked with, perhaps stretching the tech book medium even more than the graphics-heavy Head First series that O'Reilly later created. His proposals for quizzes, cross-references, and special characters to mark off parts of the regular expressions went far beyond anything our XML tools could handle and represented a puzzle too difficult for our tools team, so Friedl made his own copy of the tools and implemented his layout strategy himself. For instance, he managed to contort our tools to display a quiz on the top right page as one opened the book, and the answers on the next top right page (so they would be hidden while you read the quiz). He invented special characters and markings to distinguish the parts of the text he used for examples. The book became something of a cult classic and went into future editions, each of which Friedl did himself because no one else had his tools. Friedl stayed at my house for a while during the writing of his book, and got friendly with my kids long before he married and had a child of his own in Japan.

He offered three-hour tutorials on regular expressions at our Open Source Convention, which was still displaying the influence of Perl—the impetus for launching O'Reilly's conventions, as I'll explain elsewhere. With programmers hanging from the rafters in his presentation hall, Friedl wowed the acolytes with both deep insight and a well-cured instinct for leading an audience. He was very tall, and could have been imposing were he not always so gracious. When the cell phone of a front-row audience member went off, Friedl didn't ignore the noise or wince, but calmly announced, "You have a call." He then leaned back, folded his arms, and said in the same gentle manner, "Take your time." The audience lapped it up. Friedl was also kind to the staff at our booth, buying them a pie to thank them for all they were doing at the conference.

Every tech book has a life cycle. None is meant to last forever. So eventually, the field of regular expressions ran ahead of Friedl's marvelous book. He had written it when Perl was king of both scripting languages and regular expressions. Other languages caught up, Perl's own implementation evolved, and the field became more and more jumbled with different implementations. Friedl himself lost interest in the computer field to which he had made so many contributions, and left the field several years after moving to Japan.

But Friedl conveyed deep learning in a way few authors have ever done in computing. Many have written good textbooks about languages and protocols, while others convey enough basic skills to put users on a path where they can experiment and find their own way to mastery. But technical books don't turn readers into experts in the classic understanding of mastery that evolves from novice to expert. Here I say "expert" to mean people who intuitively know a field. Think of the term "grok" that Robert Heinlein introduced in his science fiction novel *Stranger in a Strange Land* and that became a popular colloquialism in the 1960s. Reading Friedl let programmers grok regular expressions.

Another author I worked with, Diego Zamboni, applied his virtuosity with tools to O'Reilly's. Here's the problem he wanted to solve: Many technical books contain scattered snippets of programming code. Authors usually maintain test environments to ensure that the code is correct, then copy and paste the lines of code they want into their book. Whenever you record the same thing in two places, you're asking for problems. For instance, you may fix a bug in your test environment and forget to fix the book. This problem was solved by Zamboni by enhancing the tools O'Reilly gave us so that instead of pasting in code, he could paste in a pointer that would read in the precise lines he wanted in the test environment.

One long-forgotten experiment by O'Reilly, in my opinion, held the promise to bring users to the threshold of expertise. We released some very short printed books called "notebooks" that started from the common observation that learners would rather follow a concrete example than read how to assemble their own programs or configurations out of the abstract concepts.

Authors write most traditional guides with the assumption that readers will go through them sequentially. Important abstractions are introduced in the first few chapters, supposedly preparing readers for the practical guidelines that follow.

But give the average reader a programming manual, and they will turn immediately to an example showing the task they want and copy it. Only when their version fails do they turn back to the text to figure out what they need to do to adapt the example. I have witnessed this sequence in my own hasty interactions with manuals, as I work backwards to higher and higher levels of abstraction to find a key that unlocks the meaning of the code. Although the author has diligently organized the material to proceed from high levels of abstraction to lower ones, I am too busy to care about any level higher than I need at the moment. John Dewey's concept of learning by doing describes how most programmers study.

So what did the O'Reilly notebooks do? They started with an example. Each example was then followed with a section titled, "What just happened?" The readers had the chance here to match their understanding to the subtler significance explained in this text.

The format of our notebooks reveled in their unconventionality. They were short (each under a hundred pages), with light blue covers and crude fonts reminiscent of the actual cheap notebooks that elementary school children used to carry around and fill with scribbles. This atavistic metaphor might have contributed to the indifference with which the public greeted the books. The series was abandoned after a few tries.

If no one could appreciate the sophisticated concept driving these notebooks, many readers could enjoy an earlier series along similar lines: our cookbooks. The first cookbook was written by Tom Christiansen, one of the brilliant thinkers at the head of the Perl community. He saw that many common tasks required a combination of skills and syntax elements, and created a Perl Cookbook loaded with gems on various topics. Other cookbooks followed, some edited by me. O'Reilly did not trademark the term Cookbook as it did "in a Nutshell", a phrase we used in the titles of our reference books, so cookbooks on all kinds of programming topics quickly emerged from many publishers.

Good topics for cookbook treatment are actually tough to uncover. Increasingly, the truly boilerplate software activities are embodied into libraries by leading programmers, so solving the task is simply the matter of finding the right library and the right function. A cookbook recipe that consists of just a few lines and invokes a library function does not contribute anything novel. It may have provided a convenience in the days before powerful search engines could turn up the desired library for a task, but is redundant today. Christiansen's recipes, by contrast, offered elegant models for creating novel solutions, and this is the practice of good programming cookbooks.

Later the company developed a very popular series called Head First. It was proposed by two authors new to us, Kathy Sierra and Bert Bates. They spun out in their first book a unique implementation of insights from educational research.

It is well known among educators that people have trouble retaining information from the text so beloved among aficionados of language, who gravitate toward professions such as editing. Learning springs from action, as well as from eye-catching and memorable images. Sierra and Bates evangelized a doctrine, backed by editor Mike Loukides, of engaging the responses of the human brain through low-level channels employing a roster of puzzles, games, quizzes, and images. Their Head First series had a small ratio of text to page. From the wide-angle lensed photo emphasizing a human head on the book cover to the fonts and stock retro mid-century photos, the Head First series redefined everything computer publishers had done with a book. Each concept was conveyed through many channels in order to find a chink in each reader's brain and enter there.

Expanding Sierra's and Bates's concept from their opening Head First Java book into a regular series placed a strain on O'Reilly's publishing process. Authors had to go through a grueling boot camp to learn Sierra's philosophy and her unique application of it to technical content. Editors, artists, and production staff needed to provide intensive support and use a different process than the rest of our books for online publication, which became more and more central to our business model. Head First

books used up a lot of space to offer the same basic information as more conventional books. However, they sold well enough to justify the investment.

In this section, I have been trying to show how O'Reilly threw out old publishing conventions and responded to what the editors thought our readers needed at that moment. This was rare in computer publishing. Throughout the 1980s, all publishers of computer books followed a simple three-part task division inherited from the age of mainframe computer manuals: user tutorials, programming guides, and system administration guides. The divisions between these three audiences were starting to dissolve, however.

One project in the early 1990s might be called a user manual, but one unlike any computer book that we or others had ever done: a guide to using the Internet. At O'Reilly, we were all busy using email, FTP (file transfer), online message boards, and some sophisticated experimental systems. The latter included Gopher, which presented text through a hierarchical set of menus in a crude implementation of hypertext, and Archie, a tool that crawled the Internet to find content matching search terms. We knew that researchers and techies around the world were doing what we were doing, and that more and more people were buying modems to enjoy these wonderful services at home or at work. (Modems turned digital computer output into signals that could cross telephone wires designed for human voices. More on the digitization of the telephone network in a later chapter.) Information was becoming more democratic—or so we thought at the time—largely through messages that individuals could share widely with sympathetic listeners around the world

»»

Message boards, a grassroots way to form community. existed before most people could get on the Internet. These message boards were often called news groups, but there was no actual news online in those days. You couldn't read the New York Times on your computer, unless someone typed in an article they saw in the paper and sent it around. So the word "news" came to be applied to the personal confessions—and not a few rants—by thousands of individuals seeking access to their online peers. Accuracy was no more guaranteed than on the social media that came much later. But the system was a crucial support mechanism for people lacking a voice, whether closeted gays in the Bible Belt or privacy advocates fighting corporate surveillance. The ideal of online communities as liberating environments had grown up in Berkeley and the San Francisco bay area during the late Viet Nam war era, as documented in Steven Levy's book *Hackers* and more recently in Claire L. Evans's *Broad Band: The Untold Story of the Women Who Made the Internet*.

"News" was exchanged through an ad hoc connection system called Usenet, before nearly anybody was on the Internet. People connected informally to each other using a protocol called Unix-to-Unix Copy (UUCP), where each individual's computer passed on the messages sent by others as well as their own messages in daisy-chain fashion from one modem to another.

Along with sober and professional news groups, under categories such as "biz" for business and "comp" for computing, an oddball collection of informal groups proliferated—porn galore, of course, but also serious discussions of socially delegitimized topics and marginalized populations such as recovering substance abusers, sufferers of child abuse, and gay and lesbian people. In order to make it easy for mainstream institutions to screen out topics they might find uncomfortable, all these unauthorized news items were segregated into an "alt" or alternate news group.

Although "alt.sex" was probably the most highly trafficked subgroup, many political tendencies with sometimes unbridled and irresponsible postings were put under "alt.left" and "alt.right", the latter being the origin of the name for the alt-right propaganda movement that received so much publicity during the 2016 presidential election campaign. I myself persuaded my system administrator, at one of my employers before O'Reilly, to open up access to the alt.sex group for me. This was for research on a project about censorship, not for my personal indulgence. The postings on some alt.sex groups helped me show that attempts to "clean up" the Internet were suppressing important voices.

News and multiple other services formed the subject of the book edited by Mike Loukides. Someone decided toward the end of the project to include a chapter on an obscure but fast-growing service called the World Wide Web. As a down-to-earth description of how to use the Internet, the book was simple in concept but explosive in its implications.

The success of *The Whole Internet* in 1992 rocked the publishing industry. Two other publishers had noticed trends and came out with books about how to use the Internet around the same time, but they failed to hit the sweet spot among the public. Torrents of sales and publicity poured over *The Whole Internet*. Nor did it quickly retreat into the archives of publishing history. The New York Times included it as one of their one hundred most significant books of the twentieth century.

At an editors meeting called shortly after the publication of *The Whole Internet*, we evaluated the elements of its success and made a major strategic decision...to publish more books about the Internet. This was not an obvious route to take in the early 1990s. People whose careers revolved around computers needed a strong signal to recognize the growing importance of networking. Two more signals of that nature soon came along.

Noting that the Web rapidly increased in popularity and deserved more than its modest chapter in The Whole Internet, O'Reilly was wrapping up a book on the Mosaic web browser when our system administrator beckoned me conspiratorily into his office one day. "I have something to show you," was his come-on. And on his screen was a new browser with a much spiffier layout than Mosaic. It featured the integration of graphics and text in an attractive manner that could make you think you were looking at a magazine instead of a computer screen. Netscape, Marc Andreessen's successor to Mosaic, quickly became the dominant mode of interacting with the Internet. Our Mosaic book crumpled into irrelevance upon release. The incident provided a lesson in the speed at which technology could advance, particularly given the collaborative possibilities and mass audiences created by the Internet, which frustrated efforts to cover technologies in books.

Before Netscape, the Web was text-heavy and really not much different in its experience from Gopher. Both were forms of hypertext, a concept introduced by Ted Nelson in the 1960s and already implemented clumsily by Apple Computer in the 1980s as HyperCard. (I had played with HyperCard when it first came out, and felt it limited by the tiny amount of content permitted on each screen.) Whereas Gopher offered access to distinct text sites, the Web's HTML allowed you to write an engaging, readable document and attach links to other documents to chosen phrases of your text.

It took a while for people to translate the print concept of footnotes and references into the power of linking. The new Web style focused an author on just what you wanted to say, without wasting time and distracting the reader with summaries of what other people had to say on some subtopic, instead, you would just link to their documents for further illumination or for validation of your claim. (And as Ted Nelson would pedantically point out, you'd suffer the consequences when the site you were pointing to plunged into eternal darkness.) I was soon to try out the new writing style permitted by the Web.

Netscape's support for graphics added a new dimension, opening up a Web that integrated multimedia. Yet soon my system administrator, and thousands of others, were excoriating the developers of Netscape. Its sin? Deliberately opening up to four TCP connections simultaneously in order to download graphics faster. (TCP is essential Internet software that, among other tasks, manages how quickly data flows over the network.) The early 1990s dial-up connections were straining under the load of grabbing a

picture of a few thousand bytes. The Netscape developers gamed the system by opening multiple TCP connections so as to provide as pleasant a surfing experience as they could. Users rejoiced over the speed-up in delivering web pages back then, when people joked that WWW stood for "World Wide Wait." But old-style system administrators pontificated that a well-behaved Internet application would open a single TCP connection and politely wait its turn. It would be many years before Steve Souders would work with me on a book about web performance that codified, among other tricks, the most efficient number of TCP connections to open under different circumstances.

Another indication of the Web's trajectory came in a book proposed by a talented young college student, Shishir Gundavaram, on a late addition to the HTML protocol called the Common Gateway Interface, or CGI. By adding forms to the Web, CGI expanded it yet again in unprecedented ways, because now a user could interact with the site. (Tim Berners-Lee expected users to upload as well as download content when he designed the Web, but that capability took a long time to be exploited.) Thanks to CGI, instead of just plunking down long, boring menus of options in front of the reader, a site could ask you to enter a phrase such as "sleeveless vest" and reward you with a list of relevant products.

It was a short path from CGI to hybrid systems linking up forms to databases, a trick that ultimately led to a completely innovative use of the Internet: e-commerce. CGI also made search engines possible—a feature of the Web few people could imagine living without now.

I became editor of Gundavaram's book, which quickly became one of our biggest hits. He used the supple and popular Perl language for coding. Gundavaram and I became friends and I would visit him occasionally in his Silicon Valley home as he bounced from one start-up to another. I was there to congratulate him shortly after the birth of his first child, and again when his family played out the American dream and bought a house. His brother and I visited an art exhibit in the Silicon Valley featuring tech-based installations. This was one of the first friendships that I was to develop with an author.

We already thought the Web was a pretty big deal. But it was only barely being used yet for commerce. In fact, through the first half of the 1990s, commercial use was forbidden. (So much for myths about the primeval Internet as an unrestricted medium.) Retail brought with is a whole new set of tools and languages.

⋙ It's worth looking at a few of the most popular tools of the 1990s (many of which persisted up to the time of this writing). To understand how they worked, you need to see how they worked together—and the concept of a "stack",

which helps you organize how the tools work together. The same word “stack” is also used by programmers for something deep in programming internals, but we won’t consider that here.

Right above the fundamental hardware and built-in software (firmware) provided by the computer’s manufacturer, the basic software that runs a computer system is the operating system. For a long time, the increasingly popular operating system on serious computer systems running big services was Unix. It wasn’t officially free software, but many people had access to the source code of popular Unix implementations, and Unix was treated as a standard. Linux reproduced all the behavior of Unix as free and open source software, and came to dominate these services in ways I’ll describe elsewhere. Linux is also called a “kernel”, to set it apart from all the higher-level software that runs on it.

To kernel developers, everything running outside and above the kernel is an application. The applications are granted their own computer memory in what they call “user space”, and are treated more or less as equal in the kernel’s eyes. But programmers working on the application level see subtle layers of their own, with some applications supporting others. That’s the concept behind the word “stack” as used here. We’ll look at a couple examples of how one application supports another.

Because two types of applications are crucial to web programming, we’ll focus on them here. The first is the web server, which holds the content that the web site wants to display. The web server accepts requests from clients across the Internet and sends results that browsers display. Apache became the dominant web server of the 1990s. It was developed by a loose collection of free software programmers. The organization they formed to handle logistics and legal issues, the Apache Foundation, later entered terrain far from the Web and became host to many important projects in artificial intelligence and big data. Apache is a bit heavyweight and has a huge configuration file to handle all kinds of web-related options, so some simpler web servers have intruded on the near-monopoly it had in the 1990s.

The second support application is the database, which is responsible for storing and serving up the huge amounts of data needed by major sites. Let’s say that a visitor to a web site asks for all the articles on that site that talk about free software. The web server receives the request and queries the database. For instance, the web server might look at the article database and look for the fields that hold titles and keywords. If one of those fields contains the text “free

software", the article is retrieved from the database and offered to the user. MySQL became the database of choice for web users. Like Linux and Apache, it was free software (although unlike those, it was distributed by a single company).

The SQL part of the name refers to Structured Query Language, developed by IBM in the 1970s specifically to interact with databases. It's idiosyncratic and inconsistent, and has split into so many versions that it consummately illustrates the old joke: "The nice thing about standards is that there are so many to choose from." Yet SQL is so firmly established that nearly every new database has to support some version of it. MySQL's version is more limited than many other database engines, but it provides everything web developers and small businesses want, and it's a cinch to install and start using. I edited a large MySQL series that sold spectacularly well for years.

The web server doesn't know exactly what to ask the database, because that depends on the particular business running the web site. So the business's programmers have to write small programs that run inside the web server. (You're seeing how hard it is to define layers and applications in the "stack". The business application is now running inside the web application. It gets even more complicated when we get to JavaScript.) When a request comes in from a visitor to the web server, the web server can either return a static page of information or invoke the business application. The business application in turn interacts with the database if necessary, and creates content on the fly for the web server to send out. We call this the back end, whereas the visitor's browser is the front end.

Although early web sites had back-ends programmed in Perl, a new language named PHP soon introduced itself with a bow. Dispensing with some of Perl's syntax oddities and promising a more consistent programming experience, PHP quickly took over from Perl on the Web and maintained its dominance through challenges from Ruby on Rails, Node.js, and other tailored back-end frameworks.

Now you understand the elements of web development, and the most popular examples of those elements: Linux as the operating system, Apache as the web server, MySQL as the database, and PHP as the business application language. The LAMP stack, in short. With LAMP, average web site builders could exploit the promise of interactive sites to use CGI to return content of interest to users within seconds.

Because the web developers could choose easy defaults for Linux and Apache, they focused their attention on MySQL and PHP programming. Books combining those two technologies shot to the top of programming lists. Readers prefered to start with a book tightly coupling MySQL and PHP than separate books devoted to a single tool.

I'll throw one more element of web programming at you, because of the role it played in our publishing strategy. A developer named Brendan Eich, working at the company that made the Netscape browser, made an observation that would end up changing the way we use the Web. In addition to running programs inside the web server, Netscrape realized that running programs inside the visitor's browser would also be valuable. For instance, suppose you have to fill out a form to order a product, and you forget to put your address in. This is obviously needed to send you the order. It would be nice to inform you about the missing information right away, instead of waiting for the browser to send the form information to the server, have the server's PHP program check the information, get the bad news back from the server, and redisplay the form.

JavaScript was the result. Eich made browser-side or front-end processing possible by providing a new language and asking browsers to support it. JavaScript had a superficially Java-like syntax, but despite the name was a completely different language. Programmers used JavaScript first for trivial tasks such as checking that fields in a form were filled in correctly. Once they discovered that the language gave them omnipotent access to everything about a web page, they put more and more of the page into JavaScript so that they could respond in real time to visitor's actions. The language soon became a necessity for nearly every web page.

My CGI book had been the first out on the subject, so it sold very well. Once every publisher comfortable with strings of capital letters jumped into the MySQL and PHP space, we found it much harder to draw attention to our books. I worked hard with a number of authors, but some other publishers would always outsell our books, the favored publisher and author rotating year by year. Online retail shoves marketing toward a winner-takes-all model, because many people simply choose the first option that pops up during a search. In the critical MySQL/PHP space, we needed to become that first option.

I finally conquered the top spot in 2009 with a book by Robin Nixon. A web developer as well as professional author, Nixon meticulously researched the differences in browsers, servers, and operating systems, and could be relied upon to give the most up-to-date advice. He wrote tight instructions with just the right amount of needed background, with me hot on his heels to point out forgotten information or muddled explanations.

But the stroke of genius that let Nixon conquer the field was his inclusion of another central web technology, JavaScript. He understood, as other authors did not, that JavaScript had joined the other tools as an indispensable skill for web developers. By disciplining himself carefully to cover the most important tools, and to do so succinctly, Nixon left himself room to include an entire extra programming language without making his book significantly longer than the competition. No other book included JavaScript with their coverage of MySQL and PHP. Nixon's book was number one in the category for years, going through five editions before O'Reilly management moved to different topics.

Bucking conventional wisdom became something of a competition that affected other aspects of the company besides what we chose to publish. One year we fired all our marketing staff. I remember Tim O'Reilly repeating a nostrum heard somewhere saying that "Marketing is for when you're not remarkable", and I thought perhaps that played a role. It was around this time that the business world noticed how remarkable the growth of the company was, generating a lot of positive press for Tim personally and the company as a whole. (I was featured in Fast Company simply for the novelty of creating a personal web site to promote my work.) In his review of this memoir, Tim denied carrying out a mass layoff or denigrating marketing, so this memory is my personal recollection. Whatever really happened, it left a strong impression on me because I had formed a good understanding and mutually supportive relationship with the marketing person in my area, and mourned his loss.

Actually, we knew very well that we still occupied a niche in the computer industry that was intensely appreciated by those who read us, but isolated from the vast majority of computer professionals. Mentioning our name to a stranger would impel one of two highly divergent responses: Some would utter exclamations of exaltation ("I have a shelf full of your books...") whereas others would say, "What's that?" And the latter cohort was much larger, even within the computer field.

However, a basic insight we observed about our readership stuck with us: We thrive on repeat customers. Casual computer users did not provide enough revenue to justify the cost of outreach and marketing. Among professionals in the field, on the other hand, we could find an entrepreneurial segment always seeking to learn the next promising software tool. A natural set of steps led from this observation to the creation of an O'Reilly online platform, described earlier in this chapter.

The expense of marketing an end-user book on Windows or the Macintosh could pay off if we achieved best-seller status. I think O'Reilly managers had enviously eyed the success of the Dummies series. The publisher IDG had launched that series with DOS for Dummies, and I have always assumed that the choice of the name was a somewhat fortuitous exploitation of alliteration. The marketing strategy of calling one's readers stupid, counterintuitively, hit a positive chord in the public.

Computers remain one of the few areas where an admission of ignorance can be displayed without shame and perhaps even with a touch of superiority. In the 1990s, most people had been exposed to personal computers just a few years before. Unlike other widespread technologies, computers rarely inspired love and pride like automobiles or motorcycles. Nor could computers retreat discreetly into the background and become silently indispensable, like telephones. The attitudes of the users mingled a strong desire to reap the potential that computers offered with an equally powerful anger toward the devices' bizarre interfaces and mulish refusal to do what they were told.

The love/hate relationship persisted for decades, finally mellowing into an appreciative acceptance in an age of smart mobile devices, and later voice interfaces. Still, computers will take a while to reach a golden era where their behavior is coterminous with our expectations.

The helplessness of the general population when faced with software contrasted with the joviality of those few who could grasp computer behavior and fix any flaw. So it was predictable that most people would think of themselves as dummies and assume that a book marketed as a guide for the perplexed would suit them. O'Reilly felt that we could tap into this gigantic market with better offerings than most of the Dummies series, so the company responded positively when approached by one of the most talented and prolific Dummies authors, David Pogue.

Pogue wanted to start a competing series called The Missing Manual. And thus started an unaccustomed venture for O'Reilly: a whole line of consumer-oriented books. (In his review of the memoir, Tim O'Reilly explains that he started the series to help us recover from the dot-com bust, and knew that consumer books wouldn't be central to the company's strategy.) The books flaunted a unique writing style, which I had the chance to hear Pogue describe once in a presentation at O'Reilly's Sherman Street office in Cambridge, Massachusetts. He turned certain rules on their head, such as the common injunction to avoid italic emphasis in technical communications. He loved italic, because it made the author's voice more conversational. He looked for ways to establish friendly relationships with the reader, while eschewing cheap effects that really were dumb.

Apparently the Missing Manual series filled a need, and it made us a lot of money. Pogue himself started on a trajectory to fame, weaving his expert banter in consumer-level knowledge of technology into a career as New York Times science writer and PBS film-maker.

Like the Dummies series, The Missing Manual occasionally poked its head into more arcane topics such as the PHP programming language in tandem with the MySQL database. The series wagered that the audience for this technical pairing was large enough to justify a book, because that combination of tools drove thousands of web sites in the 2000 decade.

Lax chaperones: Perl and Python (1990s)

O'Reilly has published our share of duds: books that showed promise but didn't sell well. But we have also been lucky, releasing a few subpar books that become hits. Two of our major series, one on the Perl language and the other on the Python language, were saddled from the start with books that had demonstrable flaws but were hailed as indispensable by large communities.

»»

Perl was an early favorite at O'Reilly, because it was the first language to hit a sweet spot among system administrators and was quick to learn. As a scripting language, Perl relieved programmers from tedious quality measures such as declaring all their variables (although it was so eventually added as an option to reduce errors). And despite its ease of use, it was powerful enough to write programs, particularly through regular expressions that were of unprecedented sophistication and grew only more powerful over the years. Perl also wooed system administrators by mimicking the syntax and behavior of Unix utilities they already knew, so that instead of tying together several tools with cumbersome connections known as "pipes" in Unix, they could write a few lines in one consistent, familiar style.

The historical origins of the first book on Perl, whose success went off the charts for programming books, are not really my story to tell. Nor are the reasons for its unique expository style. I'll offer just a few nods to key people in its development. Tim O'Reilly launched the project by recruiting Randal L. Schwartz, a Perl contributor so insightful that a whole Perl algorithm (the Schwartzian transform) was named after him. He in turn brought in the brilliant and gentle inventor of the language, Larry Wall.

Schwartz taught classes with a laser-focused practicality leavened with quirky humor. Wall, along with his own quirky humor, has an idiosyncratic way of finding new layers of meaning, illustrated when he was later invited to keynote our Perl conference. He found it not just amusing but also deeply significant to call his talk "The State of the Onion". And he continued to give his Onion speech long after the Perl conference expanded to be an Open Source conference and Perl was no longer the centerpiece. Although fewer conference attendees in later years turned up in sessions about Perl, Wall's thoughtful Onion talks were swamped by large adulatory audiences.

But I don't know how these two talented authors came up with the grab-bag of speculative excursions, advice from the shop floor, and other thoughts that constituted *Programming Perl*. Wall was working on the official Perl documentation at the same time that he wrote the book, and the DNA transductions between the two types of material did not merge seamlessly.

Despite the book's unprecedented and inconsistent exposition, the Perl community responded with incredible love and affirmation. Readers outside the community who wander in bewilderment through the tome complain from time to time, but their views are lost in the flood of adulation. The book sold year after year, expanding over multiple editions almost without editorial oversight. I was responsible for one later edition where I did what for me was unprecedented, and followed Tim's lead in letting the new author Tom Christiansen do whatever he thought best. Wall was not available to do much writing at the time. But Christiansen, a master of natural languages as well as programming languages, grasped the essence of what Wall and Schwartz were trying to do.

What many programmers seek from publishers is affirmation, and the mere release of this book in that period was enough to establish the importance of all the changes in computing that Perl represented—changes I will cover in the description of our historic Perl conference. To refer to *Programming Perl*, programmers would use shorthand and call it the Camel, after the animal on the cover. Many programming forums, such as the popular discussion board Slashdot, would slap a picture of this camel on items about Perl, as if that picture were a trademark of the Perl language rather than an artistic choice by Edie Freedman, our designer. Indeed, Freedman's decision to use old woodcuts of animals on our book covers branded O'Reilly more than anything we did.

Tim O'Reilly, in his review of this memoir, said, "We had a strong house style shaped by how I liked to explain technology topics, and often heavily rewrote what our authors turned in. But I realized that Larry's voice was sui generis, and surprisingly effective, even if a bit difficult for beginners. I always loved the book, and over-ruled the editors and reviewers who wanted it to be 'fixed'."

Wall writes, "Perl was designed to evolve rapidly, and we were already recognizing that the book couldn't possibly keep up with the manpages for a rapidly evolving language. So the main point of the book was to build a culture around the language. Because my training in linguistics concerned how to help dying natural languages survive through literacy and cultural self-reinforcement, I already had some idea how to get an artificial language to thrive by building a community around it. So the book itself was designed to reinforce the quirks that were already becoming evident in Perl culture. More than a set of quirks or puns, though, the fundamental purpose was to convey a certain joy in programming. I'm not aware of any prior computer language book that treated community and culture as critical language features before the Camel. "

Schwartz went on to plug a gap left by *Programming Perl* by writing a more conventional book named *Learning Perl*. I found it completely lucid. I've heard that some people are deterred by the basic Unix knowledge it assumes of the reader. But for the Unix users and administrators learning Perl at that time, the tone was perfect.

Our approach to other languages sometimes took other odd directions. Like observers everywhere, we often undervalue new technologies, especially if they seem derivative. When PHP came along, management treated it as a kind of toy language and allowed the first couple books to be handled by a junior editor who did not offer much guidance. I don't think anyone was asking the basic editorial questions: "What does the reader know at this point? Are you giving them the information they need right here? What is relevant to the task they have to do? How does your passage here contribute to their growth?"

Eventually, because it was easier than Perl to use for programming the Web backend, PHP emerged as the leading language by far on web pages' back ends, and one of the most important computer languages, all the while suffering from derision by people in the computer field who saw it as conceptually inadequate. (They did cite legitimate security concerns, and these were fixed.)

And then Python. I myself paid little attention to it at first. I had suffered through older languages such as COBOL and FORTRAN that required strict spacing and layout, so I could not see a large role for Python with its own strict formatting requirements. I also had suspicions about the lithe promises by Python lovers that it would reduce the visual complexity of code.

Python is vaunted as the quintessential easy language, free of fussiness and extraneous punctuation. True, it makes parentheses optional in many places where other languages make them mandatory, and it removes the need for curly braces by using indentation to set off code blocks. But was it good design to make indentation a significant syntax feature?

Forcing white space to be meaningful is an atavism of languages going back almost to the beginning of programming history (FORTRAN, MUMPS, and the Make utility that I wrote about in my first project for O'Reilly). No other modern language has followed Python's indentation model, despite its popularity, which suggests that language designers have considered and rejected this aspect of Python syntax.

Even when I first saw Python, I didn't think it could do much better at readability than any other language after decades of research into compilers and programming languages. As Python gets used in non-trivial real-life applications, I think my prediction has proven correct. Certain difficult abstractions, such as nested arrays of diverse types, turn up over and over as programming

languages evolve—Python just like all the rest—because these complexities are necessary for producing maintainable programs that grow large.

When a proposal about Python came our way, our managing editor Frank Willison took nominal control of the project, but not in the strategic manner that Tim O'Reilly's gave free rein to the authors of *Programming Perl*. Like the Perl book, our first couple books on Python were sprawling and disorganized. Once having found a piece of information, you could boast that the book had it—and how could the book not have it, when it approached a thousand pages in length? The real test of good writing is if a topic appears where the reader can make use of it, a criterion neglected by both Wall and Lutz.

But also like Wall's Perl book, our two early books on Python attracted large numbers of enthusiasts. They made us money, but stood in the way of our creating a coherent Python series as that language showed signs of taking a central role in modern computing. Python stands as virtually the default language for the most important developments in key areas such as machine learning and embedded computing. The O'Reilly series contains redundancies and some books that miss their mark.

Editors and collaboration (early 1990s)

I think many fields depend on a few carriers of "truth", even rich, complex fields that thrive on the contributions of many participants. Only a few attain the exalted position of truth-bearer. In the health care field, for instance, many institutions listen only to people who have achieved the status of placing MD after their names. In many religious communities, nobody feels safe resolving a debate without consulting someone who has passed through an ordination ceremony, such as a Catholic priest or a Jewish "Rebbe". Every contract has to be examined by a lawyer, and so on.

Editors at O'Reilly were long privileged to occupy this position as bearers of the truth. It was a weighty responsibility, tugging at our sleeves during every step from budgeting the upcoming year to marketing completed books. The whole company understood us as privy to a profound grasp of trends in our field and the next round of dominant technologies. We had powerful impacts not only on the planning of individual books, not only on marketing campaigns, and not even just on the direction of our company, but hopefully on the broader use of technology.

O'Reilly is now much more than a publisher—in fact, by the end of the 2010s we didn't even like to call ourselves a publisher. We became something unique where excellent content swirled together with other educational activities and tracking

mechanisms. But ironically, in the way we produced the books that we continued to release, the 2010 decade found us a much more conventional publisher than we were when we started.

Let me take you back to the half dozen years starting in 1992, when I joined the company. We had only a handful of editors. These editors were technically trained. Some had worked as professional coders, and all of us could administer our Unix system (we had only one) and fashion some Perl code on demand. We liked to pride ourselves on being different from other publishers because we were an integral part of the communities whose code we were documenting. Mike Loukides wrote a book on Unix system administration, not shying away from such highly technical topics as reconfiguring and recompiling the operating system, which many system administrators did as a matter of course in those days.

And there was no distinction between acquisitions and development, as there was at other publishers. We got to know authors in the community, perhaps contacting them on a mailing list or by surfing project sites, and we stuck with the authors once we contacted them. They could count on a single person to research the field they were in—Perl or another language, networking, or whatever (I did it all)—to work up a proposal and outline with them, to edit the book, and to shepherd it through the marketing process. This was a tremendous amount of work, but fulfilling in a total sense that kept the editors going. (Generous monetary rewards did too, but I honestly didn't think often about that.)

As O'Reilly outgrew its narrow Unix focus, it hired some extraordinary people as editors. For instance, in Brian Jepson it found a professional programmer with an insatiable thirst for exploring new technological areas and for sharing them with others. I always went to Jepson with questions on technology and where it was going. I felt flattered to see him praise my own work repeatedly, because I always felt like a novice next to him.

Jepson has worked for years on community technology forums in his home town of Providence, Rhode Island. I visited a kind of small Maker Faire event he organized there. He was familiar with software development for mobile devices and equally adept at tinkering with hardware, including 3D printing. Drawing on that aspect of his background, the company moved him into the Maker part of the business, and he eventually wandered on to other things. I don't believe the company has hired editors with such strong grounding in technology since then.

Our editors made us into the publishing equivalent of what 1980s computer programmers used to call an "engineering-driven company". This term indicates that technology trends informed what we did, and that those who made the product (in our

case, the editors who produced the books) determined the overall direction. This is in contrast to a "marketing-driven company", which could also be successful and had much to recommend it—but wasn't as much fun for the engineers.

This chapter is about "strategy", but you shouldn't imagine that we were executing detailed plans drawn up long in advance. The growth of a company doesn't resemble the football play calling systems that impressed me as a kid, so much as the tactical thrusts made by a fencer or tennis player from instant to instant. The contrast between the image of a cool planner rising above the fray and the sweaty contender holding her own on the ground stays with me as I look back over the evolution of O'Reilly.

We have never adopted a routine—there was no "business as usual". In fact, I would excuse the frustrating dearth of corporate charts and assigned responsibilities over the years, justifying it by our fluidity. True, it was a challenge to find who could carry out a trivial, everyday task such as sending a book to a potential author or providing marketing materials to a partner. But I realized that our mission, goals, and hence organizational structure was always in flux, so it would be a wasted exercise to create a clear chart listing whom to go to for what. Everybody was permanently committed to the overall success of the organization, so the right person for a job would get back to me eventually.

The company did take steps to set direction, though, and characteristically made its editors responsible for doing so. Realizing that the editors needed to pool their expertise and come to consensus regularly, Tim O'Reilly brought us together about once a year for very intense summits of the eight to ten people. I remember, for instance, the summit we held in 1992 or 1993, Ed Krol's book *The Whole Internet User's Guide and Catalog* rocketed to stardom. We had never taken on the Internet as a focus. But at this editorial meeting, we made a critical decision: We'll do more books about the Internet. Wow.

I'm making it sound like this decision was preordained, but it didn't seem like that at the time. We charted a generally correct direction at these meetings, and built up tremendous camaraderie along the way. The meetings were as engrossing—I imagine—as meetings where the Federal Reserve Board decides whether to raise or lower interest rates. Our meetings lasted a few packed days. Some of us would drive, others needed to fly. Once at the site, isolated from all distractions, we bonded, dined and tippled together, and determined how in our minds to change the course of human history through technology. Once Tim O'Reilly actually invited us to his Sebastopol house for dinner. I noted that it was fairly modest, evidence that Tim wasn't materialistic.

Apart from editorial meetings, I had the tremendous luck to get to Sebastopol once a year for many years. This was not a luxury available to most editors or most employees of the Cambridge office. The reason I could get there is that, with a minimal extra

expense, I could tack several extra days onto trips taken for other reasons. I attended conferences in the Silicon Valley at company expense, and I had two brothers living in Marin and Sonoma counties. So all I had to do was leave an extra few days or week after the conferences when booking my flights. I would pay for meals and stay with my brothers, going in to the Sebastopol office for a few days and bolstering relationships with people there.

It was particularly valuable to meet Allen Noren, who managed the web team and O'Reilly's online bookstore for many years, and Betsy Waliszewski, with whom I collaborated closely on the company's open source strategy. But I tried to drop by colleagues in every department from legal to conferences and customer service. I had the jaw-dropping opportunity to see the bustling Make Magazine lab.

Tim Allwine, a database expert employed for many years in the Sebastopol office, gave me lots of insights. Once he offered to show me a new schema for their reorganization of our MySQL database. Because I worked on most of our MySQL books (and attended the MySQL conferences, run by O'Reilly) I eagerly accepted the invitation. He took me into a room with several rectangular tables draped with long printouts covered in entity diagrams—thousands of fields in dozens of different tables. To see how large and complicated the schema was for our small company taught me a lesson in the difficulty of handling relational data. It may not be surprising that database administrators get paid so much, or that so many organizations find relational databases too heavyweight for many modern applications.

The responsibility that I felt as an editor came out in one exchange with one of my peers. Editors always develop a fondness for certain projects that don't make money. We propose some books in the hope that management will take a chance on them, and reluctantly accept the judgment that their technical superiority will not translate into sales. So fellow editor Linda Mui once told me of her sadness that a project she deeply cared about had been rejected. I reminded her that our company's income came entirely from the projects we editors led, and that a hundred other employees depended on us for their livelihood. Thus, we should see ourselves as fiscally responsible as well as visionary.

During the dot-com boom, few professionals made off with more loot than O'Reilly editors. We actually got royalties on every book we edited, just as authors did. This was pretty generous on O'Reilly's part. They were already paying better royalties to authors than most publishers. (Other publishers put impressively high percentages in their contracts but insert clauses that whittle down payments, in schemes reminiscent of those used throughout the content industries such as music recording.) An editor, on top

of that, could receive four percent of profits. I used the proceeds to expand my house and send both of my children to college with barely any loans.

Eventually, management realized that editors were draining resources that could be used to develop other departments. We were put on a conventional bonus plan, which rested on tiers of performance at different levels of the company in order to promote cooperation. First, the company as a whole had to show a profit for the year. Next, your department would have to end up with a positive balance sheet. Finally, you had to demonstrate a substantial contribution to that achievement.

I received bonuses pretty often, but couldn't ever anticipate what I'd get, much less choose projects that would potentially maximize the bonus. Cooperation and information sharing were part of O'Reilly culture from the beginning, so none of us had to change our behavior for the sake of a bonus. I ended up regarding the bonus plan like airlines' frequent flyer miles: nice to get, but not worth changing behavior for. The change in compensation pointed to a gradual trend at the company: Although content was still king (a common saying in the 1990s, attributed to Microsoft founder Bill Gates), the editors' royal status was being curtailed. Even the editors' meetings eventually came to an end without drawing remark.

☞ *Parallels always intersect in the public sphere: Activism*

Parallels always intersect in the public sphere: Activism

Contents of this chapter

Bias in artificial intelligence (2016)

Philadelphia: Cradle of American independence, cauldron of race tension, site of an enthralling collection of Rodin statues, butt of W.C. Fields's dyspeptic wit. How did Philadelphia become a strategic location for an artificial intelligence summit?

The time was October 2016, and several threads in my work came together here. I had spent most of the past couple years at O'Reilly covering the technical underpinnings of the topic we were to discuss at the summit: Artificial intelligence (AI) and the management of the enormous data resources that AI calls for.

»»

By the 2010 decade, the terms AI and machine learning described a layered approach to algorithms and data processing that conjured up computing miracles apparent to all: voice recognition, chatbots, image and face recognition, improved recommendation systems, and so on. Controversies grew up on the heels of each advance. This section explains how knowledgeable computer activists took on those controversies.

Technical guides to AI, more than technical manuals in most disciplines, must address ethical issues, especially the risks of bias. Most jurisdictions outlaw bias against what are called "protected classes" of people. Excluding a protected class—women, or people of color, or gays—can be considered an error. But this error occupies a different level from the simple miscalculations that are so common in AI, such as using statistical models that are inappropriate for the shape of one's data. This is because the latter, garden-variety errors analyze future data poorly, so they are usually discovered and labeled as disappointments by the organizations using them. But a biased model may analyze future data extremely well and be judged accurate according to the goals of the organization—but only because the model is based on past criteria, which might well be biased against one or more protected groups.

Another way to put this: Managers and staff are blind to their own bias, and their AI models may reinforce this blindness. Biased AI may allow an organization to choose convicted prisoners or other people in a manner that is very satisfactory to management, because it's just as biased as the people who previously made such decisions. (Of course, organizations hardly ever admit to basing decisions on an AI model, they call the model simple "input" to a decision that ultimately is under the control of a responsible person.)

The definition of what's biased and what isn't, the sources of bias, ways of detecting and measuring bias, and policy around all these things can be engrossing. It was a set of questions that I was ready to dive into, because questions of fairness and protection of individual rights had run through my years of work with Computer Professionals for Social Responsibility, and after the demise of that group, with the technical policy group created by the Association for Computing Machinery. The ACM, pre-eminent among non-profit organizations in the computer field, had started with a narrow focus on technical expertise and how to convey it through computer science education. At

some point they added a policy committee called USACM, later redubbed the US Technology Policy Committee (USTPC), with connections to similar groups outside the US.

When I started writing about AI, its risks had been a hot issue for some time. The muckraking web site Propublica gave the issue a very public boost in 2016 in an article claiming racial bias in a system telling courts whether a convicted criminal is likely to reoffend (information that can influence sentencing). The details behind this proprietary algorithm—which the vendor refused to disclose, even in court—would make a fascinating story in themselves. In particular, one researcher discovered that a simple calculation you could do with pencil and paper, involving just two variables, proved more accurate on historical data than the highly vaunted proprietary algorithm. In any case, after this article, investigations of bias in algorithms took off in academia and filled the news.

Around 2015 or 2016, in reaction to this research, ACM created a special group within the policy committee to examine and develop recommendations. Leaders of this group obtained funding to bring a dozen members of the group, plus some outside experts, to a rare face-to-face gathering. The group deemed Philadelphia a convenient location because it is just a couple hours from either New York City or Washington, D.C. Someone probably also had a connection to the University of Pennsylvania, which furnished the room and refreshments for the meeting.

Simson Garfinkel, I believe, came up with the decision to invite me. He had emerged back in the 1990s as an authority in computing security and privacy, had moved between several important jobs in industry and government, and had met me while proposing and writing books for O'Reilly (although I didn't end up editing any of them). When he asked me to join experts with far more experience than I in both algorithms and policy for the Philadelphia summit, I demurred at first. but he had seen me in action on other ACM projects and said something to the effect that I was good at moving people forward and getting a writing project done. I saved ACM (or was it my employer?) a bit of money by arranging to stay with cousins after the meeting.

The participants, outside of myself, were impressively credentialed. For instance, Cathy O'Neil, author of the popular book *Weapons of Math Destruction*, was invited, but had to bow out at the last minute due to illness. We had a stunning roster, even without her.

The immediate result of this highly educational meeting was a set of principles that were quickly approved by ACM leadership and publicized. I wrote up an account of the meeting and the principles for the O'Reilly web site. One of the meeting's

organizers, Jeanna Matthews, told me later that the principles received a lot of positive attention.

We expected to follow up with an in-depth research paper for the computer field's flagship publication, Communications of the ACM. I took on coordination of the writing project. But I could never marshall the experts we needed to produce the list of publications, even though many people signed up, I created an introduction to "seed" the document, and I repeatedly tried to shake the volunteers up and entice them to contribute. A few members of the group produced a short overview for the Communications, but the larger piece never appeared.

I am not disappointed by our lack of follow-through, because I believe that the field of research into AI bias had advanced at rocket speed and could not be encompassed by a simple article, no matter how well researched. The range of relevant research was growing geometrically. As for the principles, they got lost in the welter of similar lists of principles that came along from expert groups around the world. All these lists were similar, essentially taking off from well-established guidelines on privacy, autonomy, and consumer rights to enunciate rules for transparency and accountability in AI models.

I've played a similar role—writing, prodding, strategizing—for several ACM projects, and even managed to slither into a leadership role for a year on the steering committee (probably because they were short of true experts).

Working with ACM has been the most recent of the volunteer work on which I have spent countless hours over the past several decades. During these hurried years, computers became embedded in every office, every home, and eventually every device. The world moved onto the Internet and bandwidth grew to undreamt-of speeds (although not uniformly in different places). Additionally, the struggle to get services to the public over the Internet graduated into a struggle to maintain some public control over these now-ubiquitous services.

I worked on essentially all the issues raised by this history. Although this chapter presents several of them in their own tidy sections, you can take a balloon up a few miles and see them swirling about in a maelsrom with accelerating speed. My work on each topic was like a great circle on a sphere: By the premises of non-Euclidean geometry, it is guaranteed to intersect with every other circle.

Some readers in more straitlaced work settings may wonder how I found time to do all the volunteer activities described in this chapter (and many others covered in the chapter on free software, or just not covered at all in this memoir). They may wonder how my employer, or the many associates who worked on books with me, reacted to my very public shenanigans in all these causes.

To tell the truth, I never devoted thought to these concerns. I have never separated my life into compartments. All the activities in this book constitute who I am. Nor was there a meaningful distinction between the people I worked with at "work" and people I worked with on other things. They often overlapped, such as on this project where I was recruited by Garfinkel, an O'Reilly author. More than once I would conjure up a book for O'Reilly out of the relationships and topics I indulged in during what some people conventionally call spare time. (An example of a book prompted by my activism is Van Lindberg's *Intellectual Property and Open Source*, discussed elsewhere in this memoir.) O'Reilly was not only complacent but supportive while I explored all these paths, and sometimes they funded my trips. That may or may not be considered normal for an employer.

Twilight of the old guard: Computer Professionals for Social Responsibility (2012)

Like a dozen other senior members of Computer Professionals for Social Responsibility, I got the dreaded phone call in the summer of 2012. (Or maybe it was an email message—but I feel that its severity deserved a more formal channel.) CPSR was dying.

The current board of directors couldn't see a path forward financially past the next few months, and they were organizationally in total disarray. Before they threw in the towel, they gathered resources for one Hail Mary pass. (I'm no good at sports or at sports metaphors, but here I've thrown in two of them.) They decided to fly out some of their most dedicated members, the people who had given hours and weeks and years of passion and sweat to the organization, and set us to shaping a rescue plan.

Interestingly, I had never played a leadership role at CPSR. I had never led my local chapter or put in a stint on the national board. But I had contributed so consistently to CPSR that at one national meeting, a member told me, "Sometimes, Andy, I thought you were CPSR." I had written position papers, had helped with the logistics on at least one conference, and had represented CPSR to policy members—as well as allies in other organizations—for two decades. So they invited me to this final strategy meeting, and I took a plane out to Palo Alto, California.

For a long period, CPSR had politically savvy leadership who put us in the spotlight and brought in new blood. The organization came together originally around the Strategic Defense Initiative, which was popularly known as Star Wars because the movie series was fairly young at that time. CPSR activists testified before Congress against putting weapons in space, gave interviews to the press, and were widely credited with stopping the program (temporarily).

I had joined CPSR in the 1980s, after seeing a powerful advertisement displaying an atom bomb mushroom cloud with the caption, "The ultimate error message." I heard later that the radical chapter in Berkeley, California placed that ad in computer journals over the objection of the more cautious central leadership. Well, that ad snared me. And I put my heart and soul in the organization for some 30 years.

CPSR conferences drew participants from around the world, notably an annual event called Computers, Freedom, and Privacy. (One of the French attendees at that conference circled back later to invite me to a conference in France.) We spun out the Electronic Privacy Information Center (EPIC) under Marc Rotenberg, who still ran it until recently and battled hard for our human dignity. Privacy has been a constant concern throughout the history of CPSR, and my immersions in the issue proved of critical value when I got involved in health care.

My heart ached for CPSR, because I detected its dysfunction and oncoming obsolescence as far back as 2001. I heard reports on two projects at the annual meeting that year that showed me we weren't capable of turning our enormous expertise and passion into effective public interventions.

The first blow was when conference attendees discussed the recent Y2K effort. Thousands of programmers had to revisit old COBOL programs written decades before and convert dates from two-digit years to four-digit years. Dire predictions were aired throughout the press about what would happen on January 1, 2000, when programs would mistakenly interpret the year 68 as 2068 instead of 1968. Jail doors would open, dams would burst, basic services would fail. Many people, predicting the collapse of civilization, stockpiled guns and staples—I'm not joking.

The programmers came through. They fixed all the programs, and the calendar turned to the year 2000 with barely a ripple of computer problems. Some members of the public thought it had all been a tempest in a teapot, but the practitioners knew: The problem was real, and the problem was fixed.

Interestingly, I read one analysis suggesting that the convention of two-digit years was economically valid, even given the huge sums of money that went into fixing the problem. Disk space was so expensive in the early years of computing that companies saved far more money than they had to spend later on Y2K.

Where was CPSR all this time? It turned out, in our discussion at the annual meeting, that CSPR had established a mailing list about Y2K quite early on.

So why didn't we seize the publicity and hype around Y2K and put forward our expertise for the world to see? Why didn't we become the big heroes? Why weren't we flattered and feted as the authorities on this pressing topic?

We asked something like that to the mailing list member at the meeting, and he mumbled that they considered their job just to share notes with each other. I avoid indulging in stereotypes, so my apologies here—I saw his statement a classic geek response. This is why scientists haven't persuaded the public of the crisis of climate destruction. Geeks don't (in general) do marketing.

Typical geek behavior also doomed to failure our other big opportunity to have an important public impact. The topic was elections by computer.

In the wake of the horrendous failures of the Florida electoral system in the 2000 election—hanging chads, poorly designed ballots, and more—many people were calling for computerized voting machines. One shudders when reading about the many vulnerabilities in these machines. I had followed, on and off, the discussion of computerized voting on a mailing list CPSR had devoted to it. We also held a panel on the problem back around 1995. We definitely were onto the issue, and were sitting on vast treasures of expertise.

But as electronic voting machines proliferated and the public debate reached fever pitch, CPSR was nowhere to be found. At the annual meeting we asked what held back the electronic voting group from joining and even directing the public debate.

Well, it turned out that there were two factions in the working group. One thought that computer voting was impossible to secure in theory (a position that I and most educated observers have adopted), whereas the other thought that current machines were wretched but that some theoretically robust system could be found. The two groups agreed on the urgency of opposing machines as they existed at the time, but because of this rift over theory, they could never agree on a public position statement.

So I realized that CPSR couldn't be a force in the public arena. We had lots to offer but weren't offering it. Furthermore, one activist pointed out that when CPSR's experts spoke to the press or before Congress, they would identify themselves by a single institution, usually the university where they worked. The public never learned that CPSR colleagues had prepped them and perhaps set up the interviews.

Maybe I was already tired when the appeal came in 2012 to rescue the organization. But I took on the task with the discipline of a good soldier.

On the surface, the problems with CPSR were organizational. From one of our discussion documents in 2012 I quote this historical perspective: "CPSR has been most productive and successful when it had a knowledgeable executive director who worked well with the board. CPSR's decline took place during times when it had no executive director, had one who fought with the board, or had one who did not understand the organization's issues."

That was just the starting point for understanding our problem. CPSR's difficulty was its unusual reverence for bottom-up, grass-roots activity.

I've seen all these organizational tensions play out in other non-profits. The tension between professional direction and grass-roots energy has bedeviled numerous organizations that take on social policy, and the outcomes I've seen are not encouraging. Successful organizations congeal around strong professional leadership, and the grassroots efforts atrophy. Where the grassroots activists prevail, the organization meanders in a vacuum of control and eventually dies.

The loveliness of downtown Palo Alto under a bright autumn sky—we could debate and roam from our confined conference room to the sidewalk and back again—contrasted with the anxiety and gloom of our meeting. The old activists, who had supported each other through so many noble activities, bonded or rebonded afresh as we faced the sad state of the organization that formed a big part of our identities. We looked for a sliver of hope.

At the end of our appointed time, a couple days, we came up with a plan that we knew was unfeasible. We did it, I think, because after being brought to Palo Alto and housed and fed, we felt we should not just walk away with a shrug and an apology for producing nothing. Furthermore, we were engineers. Ask a bunch of engineers for a solution to a problem, and they'll provide a solution, no matter how difficult to implement it may be.

The plan involved a massive effort to rethink CPSR as an activist-led organization, with a new mission matching the difficult environment in which we were operating. The plan magnificently reflected our deep collective knowledge of where CPSR had been successful and where it fell short, it was a masterpiece of reflection on the glories of CPSR.

We put forward to the board two possibilities: To launch one final effort for our heroic and probably quixotic plan, or to shut down in an orderly fashion. Of course, the board chose the latter. I probably would have too, had I been on the board instead of on the revitalization committee.

Doug Schuler took on the task of preserving the fine historical archives on the CPSR web site, in which he partly succeeded. The rest of the work of shutting down the organization was bureaucratic or clerical—and, of course, emotional. We all had to suffer private vigils for the organization we had helped bring to a quiet end. Much of this chapter will explain why CPSR meant so much to so many leaders in computing.

Patient Privacy Rights (2009)

In the classic 1936 film *Modern Times*, Charlie Chaplin is walking along the street when he notices that a red flag used on construction projects has fallen off the back of a moving truck. Chaplin the tramp picks up the flag and shouts at the truck, waving the flag to get the driver's attention. At that moment, a large workers' demonstration comes around the corner to line up behind him, and police arrest him as the instigator.

I was often this instigator on issues of computing and network policy. I would start out as a mere observer, benefitting from my role in computing to shout out, like Chaplin, and try to compel society to recognize the importance of the issues. But my reportage would eventually turn into advocacy, and advocacy into action.

In fact, advocates for various issues have often come to me to advance their cause, granting me a leadership status I didn't feel I had. Similar superpowers were bestowed on me by many people who asked me to argue their case within O'Reilly: I didn't have the influence they assumed I had at the time—and perhaps never had such influence. Also, I was sometimes surprised when people I didn't think had taken much notice of me suddenly called and pulled me onto projects.

One such person was Gary Chapman, a noted professor in the social sciences at the University of Texas in Austin. Our paths had crossed a few times at Computer Professionals for Social Responsibility in the 1990s, but I was just a volunteer whereas he was executive director. Although I didn't think our acquaintanceship went deep, he remembered me in 2009 and asked me to join an organization he had heard about through his work. Austin was the unlikely headquarters of an organization called Patient Privacy Rights. Chapman, knowing that I had recently taken up policy issues in health care and computing, expected that I could help PPR get more publicity.

PPR was centered in Austin because it's the home of the founder, Dr. Deborah Peel, who occupied a stunning house enjoying a view across the whole city, along with her congenial husband and a daughter. Peel, a psychiatrist, became very concerned

that large numbers of patients were withholding crucial information from their doctors, afraid it would leak out and ruin their lives. Her research uncovered an alarming laxity in data collection and sharing—areas of concern that were growing in both size and risk as clinicians moved to electronic storage and new apps were tapping into customer health information.

I asked all my friends in health IT whether I should get involved with Peel and PPR. They were a bit leery of Peel, whose single-minded passion obeyed few barriers. But my friends worried not so much by her being outspoken, as by her oversimplifications. For instance, I think that PPR, like many privacy advocates, unfairly dismisses the efficacy of de-identification techniques to protect the anonymity of research data. Peel made other misstatements as well. Still, my friends felt that her voice should be heard and suggested I work with PPR.

One of the first things I felt PPR needed was more well-grounded technical advice. Peel's psychiatric background ensured a strong empathy for patients and clinicians, but she was not conversant with the intricacies of electronic records and data sharing. I talked to a number of people in health IT whom I respected to find a technical advisor for PPR, and found success beyond my dreams with Dr. Adrian Gropper.

Gropper had a stellar background. He had earned an MD but went from medical school right into developing health care technology. When I met him, he had joined other patient activists in calling for patients to control data about themselves, a policy as naturally intuitive as it is hated by health care providers.

>>> Patient data generates revenue for hospitals and clinics over and over again. Holding on to it inhibits patients from seeking out competing doctors. It can be exploited to market more services to patients (services they may not need, and that therefore inflate health care costs). In recent years, the hospitals have started to run analytics, including the AI models mentioned earlier in this chapter, to improve their own efficiency without helping other institutions do the same.

Going further than most activists to make this dream a reality, Gropp started a series of companies to make devices or software that patients could use to store and securely share data. Most if not all his software was open source—another passion he shared with me. His technical sophistication in both medicine and technology greatly enriched my work and that of PPR.

PPR was broadening and deepening at the time Gropper and I joined. Soon we were to plan our first conference, an effort that brought some 20 experts in health care, government consulting, and privacy to Austin, Texas for a planning meeting. The

conferences themselves were always at the Georgetown University Law Center in downtown Washington, DC and drew more than a hundred attendees, many from overseas. Georgetown gave PPR a generous deal.

The point of holding conferences at Georgetown was to draw policy-makers with real influence and power. The venue was about four blocks from the U.S. Capitol, so policy-makers could have attended practically by falling off their chairs. However, I don't believe a single representative, aide, or other staffer ever traveled the four blocks. Still, we attracted other political figures and felt we were gaining the ear of people we needed. As a sign of how seriously the field regarded PPR, we often got presentations from the National Coordinators for Health Information Technology and from managers reporting to them, a key department in the federal government determining rules for data use in health care. I moderated some sessions, spoke at others, and constantly blogged about the events.

The early years of my association with PPR coincided with interest at O'Reilly in health IT. Thus, my advocacy was intimately tied into my editorial tasks of finding topics, authors, and reviewers. I maintained strong ties to many people from this period, and kept writing articles on privacy and other health care policy issues even after my O'Reilly career ended.

A land of opportunity: Patents (late 1990s)

In 1995 I was invited to a strange little convocation of leaders at the intersection of copyright and technology, organized by professor Paul Jones at the Chapel Hill campus of the University of North Carolina. I had interacted a bit with Jones around these issues during my activism in Computer Professionals for Social Responsibility and related groups. I also knew a couple other attendees—author and tech freedom activist Cory Doctorow, and Pamela Jones (no relation to Paul, I assume), the founder of the free software discussion site groklaw. But I was open to meeting more folks in my areas of passion concerning policy. The most important person I met there was law professor Beth Noveck, who was just starting her assent to international renown.

At first I chatted with Noveck politely, finding little in common. But gradually, I got to see the range of her work and the all-encompassing view she brought to her world. I did not know that she had earned a doctorate in political science before getting her law degree, but the sense of inquiry and respect for history and culture implied by that achievement came through. We had lunch at the end of the conference and I was completely on board with a project she was launching, called Peer to Patent.

As the name suggests, Peer to Patent fit into the trend toward crowdsourced online participation that surfaced in the early 2000s, notably in the peer-to-peer computing technologies that I covered in a book at O'Reilly, and later also in the great expansion of user-uploaded content known as Web 2.0. The goal was to improve the quality of patents granted by bringing in domain-level expertise from practitioners in the field where the patent was being granted. Up to then, the process of awarding patents was absurdly idiosyncratic, each patent examiner—although well-educated in both law and technology—acting alone with an estimated average of 20 hours to make a decision on each application. Not every patent examiner is an Einstein. They routinely miss prior art or fail to see that practitioners would regard the patented process as obvious.

Besides the peer-to-peer connection, I quickly settled into the Peer to Patent project because I had already added patents to my study of trademarks (triggered by the domain name policy disputes I will describe later) and copyright (a necessary topic of study for anyone in publishing—and equally relevant to people interested in free software or online culture). The inevitability of my being dragged into each of these disciplines suggests that the umbrella term "intellectual property", although considered an abomination among the free software community, has relevance because people who study one of them find reasons to study the others.

Peer to Patent was also technically interesting, because it rested heavily on some experimental ways to help people find patent applications (mostly by bringing prior art to light) while keeping discussion relevant and on track. The pilot program validated the success of these collaborative techniques. Noveck and others would build on them more and more to create open government projects during the next 15 years.

»»

The computer industry was deeply divided over patents. Interestingly, software patents were a relatively recent addition to the canon. For many decades, software was considered a mental activity, neither a "process" nor a "machine" in patent law parlance. But in the 1970s, a crack in the defenses of software appeared with the granting of a patent, to be followed by a trickle and ultimately a torrent of patent applications. In the twenty-first century, more than half of all patents granted would be on software, with all the abuses and poor choices that one could guess would come along with such a chaotic, overwhelming flood of activity. As a responsible member of the computing community, therefore, I cared very much about how the patent system works.

While I tend to agree with free software advocates that patents should not be granted to software at all, I recognize the bind that both companies and the Patent Office are in. Even by the early 2000s, it was clear that more and more development that used to be built mechanically or electronically into new products was moving into software. This

> offers many benefits, such as over-the-Internet updates and accelerated testing. But of course, there are drawbacks too, such as absurd errors that no mechanical device could suffer from, and the risk of remote attack.
>
> Notwithstanding all objections, innovation is increasingly software-based. If the patent system does not recognize that somehow, patents will shrink in relevance as software grows. And we'll have to think about how to reward innovation without patents. Meanwhile, we have to make sure patents are granted on true advances, not cheap land grabs in various areas of product development.

Within two years of hooking up with Peer to Patent, I had ridden its momentum enough to break into two journals where I had never imagined I could earn the honor of writing: The Economist and Communications of the ACM.

The gears of the universe happened to click into place at just the right time for me and The Economist. I happened to have communicated with their tech correspondent. I wrote to him and got attention for a proposal that I'm sure would have gone right into the trash if I had pursued normal channels. He got permission from higher-ups for me to cover Peer to Patent for a special issue called their Technology Quarterly. It was for this article that I thought up the sentence, "Not every patent examiner is an Einstein," but I deleted it under duress after the straitlaced reviewer at the Patent Office felt it to be insulting.

I lavished more obsessive detail on this article than any other piece I've done, at least on pieces that are more expository than creative. Not only did I contact a wide range of people at many companies and institutions, but I got a couple books on the subject (one recommended by Noveck). Furthermore, I read several weeks of the Economist to absorb their distinctive style, which includes a perplexing, ironic linguistic zigzag that the upperclass British like to call humor. I even analyzed the journal's grammar and set the spell-checker on my GNU/Linux system to British usage so that I could spell words as they expected.

My article appeared without a byline, the normal Economist practice. What mattered to me was the thrill of introducing Peer to Patent into this influential outlet. The article came across in a big way, as a two-page spread. Noveck (who of course had reviewed my draft) saw the publication before I did and sent me an email message with the one-word subject WOW.

For Communications of the ACM, the most prestigious journal in computing—at least among journals with a general scope—I also did significant research. I learned how to read a patent, a descent into a special hellish place all its own. Earlier, Noveck had asked me to write an article to recruit reviewers for a particularly flimsy patent application: one that vaguely tried to grab ownership

of an area of user interface design and wait for someone else to do the hard work of actually solving the problem it tried to own. I analyzed the patent and explained its deceptive presentation in the article.

My article for the Economist explained the purpose and justification for Peer to Patent, on a policy and business level. In contrast—because I never write the same article twice—my article for Communications of the ACM was aimed at actually recruiting computer experts to volunteer for Peer to Patent. As part of my page-and-a-half piece, I explained my technique for reading a patent application.

Besides the extreme obfuscation and fragmentation found in most patent applications, they make reading difficult because the text refers to many figures, but the figures are not inserted in their proper places as in a good technical document. Instead, the figures are gathered in a separate place, which I guess made sense given the printing capabilities of the age when the patent system was invented. When reading the application online, I found that the trick is to open two browser windows, one for the text and one for the figures. Then a reader can see how they correspond.

Not only did Peer to Patent extend my writing to new venues, it gave me a chance to interact seriously with Wikipedia for the first time. It was clear that Peer to Patent needed a Wikipedia page, as an historic social and technical experiment. I decided one day to write up a page from scratch. I couldn't offer much detail, not being a member of the team, but I offered valuable background under headings such as "Justification and purpose" and "Theoretical underpinnings". After watching the new page go live, I wrote to Noveck saying I had a present for her. I remember her being both flustered and flattered by seeing the Wikipedia page. She told me her team had been talking for months about creating one, but no one could take the initiative. What can I say? Sometimes writing calls for a professional. I think the conventions of Wikipedia were simpler in those days, so I could get the editors to accept my page without much arcanery. Twenty years later, the page still showed the basic structure and some of the text I put there the very first day.

Eventually Peer to Patent wrapped up with gratifying success: The Patent Office incorporated it into their routine process. Noveck's reputation soared with an appointment to a two-year White House position, the publication of a couple books, and the spread of her ideas worldwide. She embarked on a set of jaunts to places as far as Russia—yes, they showed interest in government transparency—and Hong Kong.

I was reminded a decade later of the craziness inherent in the patent system. A patent lawyer called me and asked for testimony related to a case he was handling. It all stemmed from my 2001 Peer-to-Peer book, which was oddly appropriate because

the peer-to-peer movement, with my book at the center, had partly inspired Peer to Patent.

Here's the story. A year or so after Peer-to-Peer was released, a patent troll tried to patent technology that had been documented in that book. At that time, other companies challenged the patent application, but the office said that the book had come out just a couple weeks too late to apply. The book's release date was March 2001, and the troll had managed to submit the patent application just within the one-year deadline that allowed them to claim they had really invented the idea. This is how the patent system works.

Apparently, the patent was still relevant fifteen years later. A new patent lawyer was brought on the case, and he hammered on the issue with more sleuthing than the earlier lawyers. His dogged research turned up a blog posting I had written (once again, my blogging proved valuable both to me and to others) where I mentioned that we brought early copies of Peer-to-Peer to our February 2001 conference in San Francisco. This odd little detail, stated in passing, proved to clinch his case. Because the conference was open to the general public, offering 140 copies there constituted "publication", and the couple extra weeks it subtracted from the official publication date put the book within the scope of "prior art" that could overturn the patent.

I sent him a copy of the book along with a signed affidavit affirming that the book had gone on sale in February 2001. I didn't hear back about the court's final decision, but the lawyer was very happy.

This little incident, besides illuminating some odd corners of patent law, shows the value of keeping old articles online indefinitely. Here, we may have righted a wrong and championed innovators just by preserving a fifteen-year-old web page. What has the computer field lost with all the pages that the O'Reilly web staff wiped out? (If you feel that I'm harping too often on the lamentable destruction of legacy, consider that anyone writing a 200-page memoir must consider the preservation of history important.)

A picket, a packet: Telephones and the Internet (late 1990s)

In intensity, longevity, and perhaps social significance, the biggest tech issues I have dealt with surround telecom policy. Most people can think of no topic so boring. A few ears will prick up when I tie it to terms that have garnered a lot of discussion: "network neutrality" and "high bandwidth". But really, I was a foot soldier in the battle to create the world we live in.

People everywhere pull out their phones before going somewhere or buying something, hardly even considering the web of fiber and cell towers that makes this interaction possible. In the 1990s, few could envision today's communications world. And nobody could achieve these connections without ubiquitous Internet coverage.

Yet this marriage between mobile and the Internet—the subject of many tedious screeds in business and technology—is no utopia. Connection costs are high, competition is low, and availability is grossly uneven. The COVID-19 crisis has finally forced governments to admit that they've let their constituents down, creating digital divides geographically, racially, and economically.

The problem is that, as networking started to show promise in the 1990s, public policy-makers left all the decisions about deploying these networks up to private companies, which based them on cold business calculations. The lock on Internet service is relaxed by special subsidized accounts for low-income residents and other universal service measures, but a digital divide remains entrenched. (One FCC chair dismissed any reponsibility for this problem with a joking complaint about suffering from a "Mercedes divide".) Now these companies are angling to lock in the digital divide with 5G phone towers, which are technologically constrained to serve dense, affluent neighborhoods and leave poorer, more rural areas in the dust.

That's the story of this section, which I hope you will no longer see as boring. The public is waking up to the social implications of telephone technology. What's interesting is how long it took computer technologists to discover it.

⋙

As ludicrous as it now seems, computer technologists up until the 1990s thought little about networks. Computers then were largely stand-alone systems. Connecting the computers was an afterthought. Administrators did tussle with networking, but they busied themselves with the details: First stringing cable for local area networks, and then configuring the ever-expanding list of protocols for connecting at the software level. They were kept busy fussing over the software, because every protocol was inadequate and could be fixed only by introducing a new protocol on top of it, simultaneously introducing new needs for which the new protocol was inadequate.

The administrators contracted with long-distance telecom companies, but didn't investigate much what those companies were doing—except for a few like Barton Bruce, an eccentric wizard who wandered in and out of the computer room at O'Reilly's Sherman Street office. At that early stage of the company, we were sharing the office with a computer consulting firm. Bruce, whose relationship to the computer consulting firm I never discovered, held mysterious seances with the Internet in our large, glassed-in server room. Whenever he wandered from that sanctum

to get a drink of water or perform other needed activities, he'd stop and chat about the monopolistic practices of Tier I providers or the causes of bottlenecks at the MAE-East exchange point. The cheerful upward tilt of his busy mustache became a warning beacon. People would discreetly find a way around him and the ruminations he loved to let loose on anyone lacking an easy exit.

I awoke to the fundamental role of telecom in the mid-1990s when Congress began to discuss a major reform bill. The telecom industry had chugged along quite nicely since the 1982 consent decree broke up AT&T and subjected it to competition. MCI and Sprint jumped in to create new long-distance lines. At some point in time I cannot establish, each launched a line of luxury personal devices called mobile phones. Few companies tried to compete over the "last mile" of local phone service, because that would have required spending billions of dollars and getting thousands of local permits to string more wire in the neighborhoods they wanted to serve.

So there were reasons for legislators to rethink the telecom space, even aside from the elephant in the room that was rapidly evolving into a whale: Internet service. I won't detail the many tiresome ways that telecom companies opposed the Internet and put stumbling blocks in its way, up until they earned enough money (mostly from traffic created by the Internet) to switch strategies and swallow the Internet. I've written exhaustively about that history over the years. But in the 1990s, it was critical to extend Internet service to everybody at a reasonable cost. A diverse set of issues drove the discussion that ultimately congealed into the 1996 Telecom Act.

As I delved into free software and Internet technologies, I found in telecom a whole new and bewildering area for study. The more I understood the issues driving telecom competition and its effects on freedom of the Internet, the more parallels I saw with free software.

Exploring the telecommunications infrastructure that lay underneath everything people were doing (literally underneath—in fact, laying cables in the ground was one of the hot issues) led me to see everything anew. I developed fascination for those little finance charges with meaningless names that appeared on monthly bills. The most obvious sign of my dangerous descent into wonkdom was a prurient interest that I started taking in those long discursions of Barton Bruce that everyone had been avoiding. He gave me some background into telecom and Internet service as it was in the mid-1990s. But that background was soon to utterly change.

The best gathering place for people like me, on the cusp between telecom and computer networking (or as practitioners would say, the Bellheads versus the Netheads), was Computer Professionals for Social Responsibility. An incredibly dynamic and perceptive technologist named Coralee Whitcomb ran these efforts. Whitcomb had gotten into computers after entering a Boston-area business school, Bentley College, for her undergraduate degree. Her advisor told her that she had a choice between two areas: accounting and computer science. The latter sounded more interesting, so she leapt into what turned out to be a maelstrom.

In addition to staying at Bentley—which grew from a dinky business school to a much-sought destination for learning—to teach computer science, Whitcomb tirelessly launched one social project after another. As expressed by Steve Miller, another activist with whom I worked closely in CPSR, "Coralee goes and does the things the rest of us sit around and talk about doing." She opened a computer training center for the underprivileged, Virtually Wired, in a downtown Boston storefront. She ran a cable news program about Internet issues, where I once spoke about the Communications Decency Act (a toxic insertion forced by right-wing legislators into the Telecom Act). She organized conferences.

And Whitcomb directed CPSR's ground-breaking telecom work. She shuttled between Boston and Washington, getting to know everybody important on telecom issues as well as the technical and legal complexities we'd have to contend with. I think CPSR had some positive effects on the final bill, certainly in the parts about universal service, and perhaps also regarding competition.

»»

The Telecom Act recognized the value of having more competition to increase innovation and lower prices. For instance, to help new companies enter markets dominated by the old monopoly telephone companies, the law enabled the FCC to determine 14 "interconnection points" where the incumbent telephone companies had to allow exchanges with their competitors. It turned out that only one or two of these interconnection points were useful, and the incumbent companies continued to find ways to slow down interconnection.

Of the many schemes used by the incumbents to avoid competition, I'll mention a relatively simple one: hoarding phone numbers. If you wanted to switch companies, you had to give up the phone number you had given out over the years to friends, family, and business associates. Thus, "number portability" became a rallying cry for new companies. Eventually, the government made it possible for a person to keep their phone number during a switch.

How could I let languish such a fertile and pressing concern as telecom policy? The very core of the Internet—the goal of allowing access without hindrance—was at stake. Dozens of articles I wrote over the next twenty years delved into the myriad issues

around telecom, winding their away among the issues of competition, cost, universal service, and Internet freedom.

In those days we thought we could achieve all good things in the Internet space by ensuring competition. But we didn't reckon with the sheer scale required by telecom. The large companies just got bigger and bigger, then merged, and merged again, to get enormously bigger.

Some people in my activist circle tried to adapt to the new situation by calling on the government to provide wires and Internet service as a public good. I think this can be useful in particular situations (I'm sympathetic to municipal networks), but I tend to like the old-fashioned and frequently undermined free market for technological development. I've seen too many government policy decisions that end up favoring powerful companies, skewing technological development, or just trying to gain control over everything.

Other activists, frustrated by the ongoing takeover of Internet service by large corporations, call for heavy-handed control to keep them honest. The term "network neutrality" running through these proposals didn't exist back in the 1990s and early 2000s when we had a shot at really creating a healthy telecom industry. Although I could support some limited oversight, I think that technology is hard enough to get right without a company having to twist its decisions (such as traffic shaping, which is crucial to providing good performance) to meet some tangled regulation.

The days of the mom-and-pop Internet provider, friends to everyone they served, were mostly doomed. A few may still survive in isolated areas that the big providers are too aloof to serve. Many local providers put in heroic efforts to keep Internet service going, just as the stalwart George Bailey, played by Jimmy Stewart, provided banking services to the people of his town in the movie *It's a Wonderful Life*.

One of the people who always inspired me was Brett Glass, who offered wireless Internet to the population of Laramie, Wyoming. He was never shy about leaping on a roof to install an antenna. Glass expressed contrary but carefully argued positions about a number of things, including network neutrality. He also strode boisterously into matters of free software. Glass was a big user of BSD, a clone of Unix with great historical weight that lost its momentum to its competitor Linux. His reddish beard was a warning and portent to any conference where it appeared. Many people resented his aggressive presence and criticized him for speaking his mind, but I listened carefully to his reasoning and always learning something from him.

I wonder how Coralee Whitcomb would have treated recent developments in telecom and the Internet. She did not last long enough to see current events, coming to a tragic end because she was a year late getting a necessary breast exam. After a bunch of us from CPSR came to her farewell at Bentley College, she died of cancer in 2011. Scandalously, she appears nowhere in Wikipedia, evidence of the diminished regard given to women there. She deserves a page of her own. In fact, I tried to start one—an effort backed enthusiastically by our CPSR friends and colleagues—but it proved an impossible task, because the relevant source materials that would document her many contributions to life in computing today had been recorded either in ephemeral paper form or on long-defunct computers, and weren't retrievable.

The career I didn't know I had: Journalism (1997—1999)

From February 1997 through April 1999, I wrote 300 words weekly for the American Reporter. I served as the Internet correspondent for this scrappy little online newspaper, and flexed my muscles as both journalist and advocate in all the policy areas that interested me. The pace was bracing, but I was always turning new corners and finding treasures beyond them. Internet journalism was an adventure game.

The American Reporter was an early experiment in online news, more disorganized than organized by its founder, a veteran of journalism named Joe Shea. The 300-word maximum was an arbitrary discipline imposed by Shea, perhaps because he measured all his writers' contributions by word count and promised us equity in the operation if the American Reporter ever turned a profit. Shea offered stories for sale to other media, a business plan that might have made sense in the 1990s. But I don't think any of his contributors ever expected to make a penny. We wrote either out of our personal regard for Shea, an oddball in his own manner, or because it provided us with a forum for our unredacted views—certainly the main draw for me.

What did I find to write about each week? This was never difficult. The world was changing in every way, pressed onward by activities on the Internet, and the mainstream media was oblivious to it. I needed to tell the story, because few other outlets were doing the job. And I had feelers out on every topic: Internet speeds, pricing, threats to free speech and privacy, online organizing—you name it. My connections crossed borders and oceans, I could just as easily report on events in France, Germany, or Peru as in the U.S. I could also read a few languages well enough to check primary sources. My main task was to decide which of many controversies to highlight each week.

I never posed as a mere reporter—every piece I wrote had a definite point of view. But I insisted on precision. One security expert told me that he granted me an interview because he was angry at the misinformation that the mainstream press was promulgating about a particular issue, and saw that I was one of the few authors who took the time to get it right.

In addition to my commitment to tight deadlines, Shea also needed someone to administer the web site, which was hand-coded in an obsolete version of Perl. Knowing that language, I agreed to take this on, only to discover that the person putting the site together was both the worst designer and worst programmer the world had ever seen. The design was so rigid that new browsers would break the layout, and I spent a good deal of time accounting for the oddities of different browsers.

>>> After hours of dogged examination, I figured out that the bizarre code processing Shea's articles (they had to be converted from plain text to bad HTML) hid an arcane state machine. Any variation in input, such as a missing space or line break, would cause the entire run for the day to fail. The designer had made a weird use of frames to hold graphics and text. I fixed all the problems but didn't dare try to make any improvements because of the thoroughgoing fragility of both web design and state machine. I did help Shea incorporate advertisements, counters, and new menu items. He gratefully dedicated a menu item to my writing—the only contributor to get their own place in the drop-down menu.

Despite the web site's drain on my time, American Reporter was fun and fulfilling. My publications there led to other opportunities in both Internet activism and journalism. I stopped only because I felt the need for my commentary on Internet policy had receded.

When I began writing for the American Reporter, mainstream coverage of the Internet was characterized by TIME Magazine's notorious 1995 cover displaying the word CYBERPORN and a child whose face, bathed in the light of a computer screen, breathed a mix of fascination and horror. News "reportage" like this fed right into the Communications Decency Act of 1996, a naked bid at censorship that the ACLU and many other organizations had to combat fiercely. As antidote to conventional approaches, I saw it crucial to offer my thoughtful research and analysis about online speech, telecom regulation, universal access, copyright law, and other serious issues affecting the Internet.

But a few years later, the media started to do their job. The New York Times (particularly in the journalism of John Markoff), the Wall Street Journal, and other major publications discovered the Internet and were lavishing on it highly professional research I

couldn't match.

Although I would continue to administer the crotchety American Reporter web site until Shea went into the hospital for his final decline, and would write occasional articles I thought his readers would enjoy, I switched to a less demanding but more visible position through my monthly Platform Independent column in Molly Holzschlag's Web Review. That lasted until the dot-com bust put the kibosh on Web Review at the end of 2001. For Holzschlag I wrote some of the articles with the most lasting power, including some studies of peer-to-peer and a parody of Dickens' Christmas Carol.

"The Ghosts of Internet Time" went up on Web Review just before Christmas in 1999. Here I drew on themes from Dickens's famous story to provide a view of Internet history and issue a highly idealistic call for action to preserve its best aspects, all in 1100 words. More than 20 years later, I still see pertinent insights in the words spoken by the Ghosts of Internet Past, Present, and Future.

For some reason, "The Ghosts of Internet Time" captured the love and imagination of people around the world. A few years after its original publication—when Web Review had already folded, I think, and I was hosting the story on my personal praxagora.com web site—someone wrote to say they wanted to translate the story into another language. This opened the door to a flood of translations, all by volunteers. About 20 are still extant as I write this. Some of the volunteers were acting out of pure altruism, it seems, while others wanted the content on their web site to promote their commercial activities. I welcomed them all, and trusted that the translations into languages I didn't know were faithful to the original.

"The Ghosts of Internet Time" was representative of a stream of articles where I pondered the future effects of technological chance. One of my early blog postings, from 2000, is titled "Dialog with an Internet Toaster", showing that I was already thinking about the Internet of Things and the trends that O'Reilly would cover in its Solid conferences fifteen years later. I have used stories and skits many times as concrete analogies for my ideas about computing and the Internet. Another little parody I put up in 2000 suggested the theme of patients sharing data to help each other find a cure, anticipating a call for patient control over their medical data. Later short stories predicted the breakdown of journalism ("Validators", 2007), the triumph of cloud computing ("Hardware Guy", 2010), and universal social tracking and rating ("Demoting Halder", 2012).

No fading signal: Voice and video (mid 1990s)

In the mid-1990s, Internet researchers were groping toward real-time, interactive communications—what we now take for granted in services such as webinars and Skype, not to mention the virtual conference tools such as Zoom that became necessities during the COVID-19 pandemic. Just as search engines went from a nice-to-have service to a part of core Internet infrastructure in the late 1990s, virtual conferencing made the same transition over just a couple weeks in March 2020, as countries everywhere imposed physical distancing on their residents.

Videoconferencing stems from the courageous and visionary innovators—yes, those strong words apply— who implemented interactive phone calls and video sessions on the Internet in the 1990s. I don't believe this part of Internet history is told widely enough.

»»

The obvious bottleneck to Internet phone service is low bandwidth. Also implicated is the brilliant core insight of the original Internet as created in the 1960s: packet-switching. The whole idea of the Internet rested on its radical process of breaking communications into small chunks and sending them over potentially many different paths to be reassembled at the end. This worked exquisitely for communications with no dependencies on time. When people tried to use the Internet for real-time communications, they suffered from jitter and lost packets.

Overcoming these problems required clever engineering. The emergence of the Internet as an interactive medium drew together the efforts of many inspired geniuses. I learned about the subtle interactions among these technical explorations from one of the people who shaped popular computing, Bob Frankston. He helped to invent VisiCalc, which in the infant years of the PC era showed the public what computers could do for average folks. He also introduced Microsoft to home networking, then got into telecom policy like the rest of us in the 1990s. Frankston helped me understand that low bandwidth could be overcome through improved compression and other aspects of sophisticated communication protocols. He was never a cheerleader for higher bandwidth as a solution to technical or social problems. He wanted network engineers to make smart use of the bandwidth they had.

Because Frankston lived in the Boston area, I had the privilege of meeting with him several times, sometimes for Boston-centered conferences and other activities, and sometimes at the O'Reilly summits known as FOO camps. Frankston came to rely on me to bounce off his ideas about telecom reform. With some trepidation, daring to summarize his multi-layered views on the topic, I can summarize his approach as follows: He wanted to move Internet access from a business proposition to a core part of society, like

streets and sidewalks (a metaphor he regularly used). I assume, but can't guarantee, that this meant a call for declaring Internet access a government-backed utility, as many other Internet activists have done.

Frankston regularly sent me articles for review, sensing apparently that I could help him reach a wider public. His mind tends to unique associations, a trait I could appreciate because I think I have a highly unconventional organization to my own cortex. I usually read his drafts elliptically, rotating them in my mind and pinpointing one or two key points, then asking him to hang everything off those points.

Providing telephone calls over the Internet became a kind of messianic striving in the 1990s. One of its greatest champions was an entrepreneur named Jeff Pulver, who consulted with businesses hoping to exploit those voice services. He created a series of conferences called Voice on the Net. As bandwidth improved, along with protocols, Pulver renamed his organization Video on the Net. Everyone who wanted these efforts to succeed would gather for human networking and business deal-making at these conferences. The legacy directly informs the services we enjoyed in the twenty-first century. Another enduring feature of these conferences is the Video on the Net briefcases, one of which I was still using 25 years later to lug my laptop on trips.

One group of researchers spearheading the interactive Internet felt that their innovations were so radical that they deserved the name "Internet 2". I wrote about their research in blog postings, and once they asked me to deliver a talk over their video connections. I took this on with great enthusiasm and went down to a studio in Boston where they transmitted my talk. (The exact topic escapes me.) I exhibited great energy during the talk, accompanying my words with grand movements of my upper body that I thought would convey energy. Viewing the video later, I realized that all I had accomplished with these movements was to introduce distracting and unseemingly jitter. I needed to learn a lot to use the Internet medium effectively.

The vision of Internet phone and video calls was politically toxic in the 1990s and early 2000s. Telephone companies, newly spun off from the AT&T breakup, derived huge revenues from charging 50 cents or a dollar for long-distance calls, and truly punitive rates for international calls. They regarded the prospect of people making free calls as an existential threat, and fought it by persuading regulators to ban or place insurmountable barriers in the way of the new, small Internet carriers.

Just to give an idea how this battle was fought, one of the big issues revolved around emergency services. It is certainly a great benefit for emergency responders to know the location of a call when someone calls an emergency number such as 911. Landlines provide the information on the spot. But computers connected to the Internet make it harder. So what happens if someone uses the Internet to call 911? Telecom companies spun enormous controversies over this possibility, trying to rule out Internet phone calls because they didn't offer the same guarantees as landlines. This seems particularly ironic years later as the same telecom companies degrade and eliminate landlines in areas where their revenues are dropping.

And of course, the telecom companies were making money hand over fist from the Internet all this time. When telephone lines were slow, many households invested in a second line to support their Internet logins. In general, explosive increases in Internet use during the 1990s and 2000s led to explosive increases in the use of telecom lines, which in turn drove lusty telecom profits. Internet activists tried to point this out in our parries to the assaults that telecom company lobbed at Internet use. Eventually, the Internet's offerings proved so enticing that the telecom companies had to embrace it—and ultimately try to take it over. They had also learned by then that they could radically reduce their costs by digitizing their own lines. It used to be that computers hooked up modems to send digital information over phone lines meant to carry voice traffic. Eventually the technology flipped completely: People's telephone conversations were digitized to go over lines using the Internet's basic concept of digital packets.

Voice and video on the net were not inevitable developments of the telephone business. They are the legacy granted us by determined experts and advocates like Frankston and Pulver. Everyone who hops on a video session for a class or a work meeting should honor these pioneers.

Audacious in deed: Cyber-rights (mid 1990s)

I joined CPSR as a novice in Internet policy, jumping into vastly complex areas such as telecom policy with the aim to learn. But I gained most of my confidence as an interpreter and explainer of Internet policy after I volunteered to help moderate a CPSR

mailing list called cyber-rights.

We invited the public to join us in advocating for a set of four rights, three of which were fairly unarguable: the right to assemble in online communities, the right to speak freely, and the right to privacy online. But the fourth, the right to access, turned out to be controversial.

One might think that the right to access was kind of essential, because one could not enjoy the other rights on the Internet without access to it. Indeed, that right informed the central issue of telecom reform that CPSR took on during the mid-1990s. With the COVID-19 pandemic, institutions in 2020 have come to complete accord that every student, every unemployed person applying for aid, everyone needing any kind of connection to the greater society, needs Internet access. But in the 1990s, a significant minority of CPSR members and supporters had a libertarian bent, which is fully in line with free speech and privacy, but frowned on the subsidies and regulation implied by our call for universal access.

As if to draw a line, our web page introducing the group in 1995 started "The most important civil liberties issue facing us today is getting citizens of all races, classes, and creeds connected to the Internet. Our fight for free speech and privacy rights will remain a hollow victory if cyberspace remains mostly a bunch of well-educated white folks." I suspect from the clarion-call insistence of the rhetoric that I wrote that paragraph.

Even though the list moderators resolutely agreed on the principle of universal access, we found ourselves eventually in dispute over how to achieve it—and that was fatal to the list.

As relevant events passed faster and faster by us—position papers, proposed regulations in many countries, technical achievements, occasional media coverage, and our own activities—I thought it helpful to write short summaries for the list. I was effectively acting as a news correspondent, and this eventually led to my being one for real. Joe Shea was on this list. He had quit his day job to try journalism on the Internet years before any mainstream publication made the move. He wanted a regular Internet feature, and recognized me as the person to do it. I have described my adventures there in an earlier part of this chapter.

We activists were all jazzed up by the outpouring of personal narrative and activist engagement on the Internet. I captured some of this breathless feeling, the anticipation of crossing a continental divide into a new land of freedom, in "The Ghosts of Internet Time". I flaunted the Internet's liberatory potential even more shamelessly in a 2001 short story, "The Meaning of Independence Day". Enfolding a lecture on free speech and censorship along with a high-jinx adolescent adventure, the story ends

with an ambiguous exchange between the heroine and the school principal I set up as her epic antagonist. The principal cites the Internet as the first mass communications platform that will challenge entrenched powers, and I leave the reader wondering both what the principal really desires and whether the heroine can win the battle to which he invites her.

If you're reading these sunny reckonings amidst the wreckage of democracy caused by bots and malicious analytics run amok, and conclude that we were naive in the 1990s, let me take strong opposition to that view. My colleagues in CPSR did not believe that liberation was guaranteed. We poured our free time into policy campaigns precisely because we knew how easy it would be for our rights to slip from our hands. The poison on the Internet lies not in the contributions we celebrated from the masses, but in the vast machinery of data crunching that slots people's attention into convenient boxes for marketing.

So let's talk. Do I still nurture a trust in online media to bring people together and channel their strivings to save the world? Yes, I do. In fact, had I not maintained that faith, I could not have devoted the bulk of my waking moments over the past 35 years to explaining how digital technologies work, encouraging professionals and lay people alike to make the most of these technologies.

Back to the mundane grime of life as an activist, and the demise of the CPSR mailing list. Having multiple moderators was a terrible mistake. We started out joyous and positive-minded, but divergences appeared after a few months.

One moderator was a telecom professional, with a background, I believe, in law. His expertise was formidable and very useful in dissecting federal policy and planning our campaigns. But he was cautious in his policy proposals, knowing how the current industry was structured and how difficult change would be.

Another moderator was an artist with a great zeal for our cause. His approach to our universal access principle was visionary. He looked forward to ripping away the artificial barriers of current telecom behavior and policy.

Both approaches have value and integrity. Both were worth hearing. But as the two moderators' views started to clash, debate became acrimonious. The professional poured scorn on the artist's naïveté while the artist accused the professional of being a sell-out. Neither list member could be barred, because they had the status of moderator. The medium of email, notoriously conducive to exacerbating disagreements, did not permit a healing sit-down negotiation that might find common ground. I suppose I could have tried to take the dispute to a higher level of CPSR, but I didn't know what that level might be—we were not a strongly hierarchical organization. I tried to cool the two combatants down, but to no avail. Members of the list drifted away, and finally, in exhaustion, we killed it.

Domain names: A question of Internet governance (mid 1990s)

The world of the Internet was evolving rapidly in 1994, and as a novice activist I was putting out feelers to find topics with which I could grapple, express myself, come to terms with the tectonic changes taking place, and perhaps leave a mark on events. The first issue to come up on this beat seems odd even today.

One of the colleagues with whom I'd struck up a casual friendship was a lawyer. I probably met her through Computer Professionals for Social Responsibility, but I cannot guarantee that, because those of us concerned with Internet policy were a small and tight-knit community and could run into each other anywhere. This lawyer—whose name I unfortunately forget as well—suggested I explore current controversies over domain names.

>>> Of course, I knew a lot about domain names. I had reserved praxagora.com for my personal web site, so I understood domain names from the point of view of a proud owner. I also understood the technology, because if you enter praxagora.com into your browser you are tying into a vast and rigorously maintained system called DNS (which some say refers to Domain Name Service, others to Domain Name System) whose operation must be understood by system administrators.

> In 1994, there was no comprehensive, consistent, or coherent policy regarding the allocation and management of domain names. The closest thing to a controlling body was a single person, the redoubtable Jon Postel. A company with unclear and idiosyncratic provenance called Network Solutions controlled all the important top-level domains. A lot of Internet activists wanted some policy assuring that the public interest would be represented in domain names.

I remember telling the lawyer, "Sounds like pretty small potatoes to me." "Not at all!" she replied. If I knew that the next five years of my life would be filled with meetings, blog postings, position papers, and other political activism around this topic, I might have walked the other way right there.

I don't expect every reader to care about domain names—although I know that some do. I met someone at a reception of a Harvard University forum about law and technology who pummeled me verbally with condemnations of how domain names were administered. But for most of us, the more interesting issue is the broader philosophical question raised by domain names.

»»

In retrospect, the fight about domain names seems inevitable, because it's a battle over centralization of control.

The Internet is famously decentralized. I think the story that it was designed in decentralized fashion to survive a nuclear attack is at least partly true, but in any case it resolutely lacks a controlling authority. During its early years, each technical advance would be adopted or rejected by each hub running Internet software.

Various countries ranging from the People's Republic of China to Saudi Arabia and Russia have imposed controls, but they do so either by punishing actors who are out of favor after the fact or by inserting firewalls at large Internet service providers. The network does not in itself support central control, although once in a while a huge company such as Google could essentially say, "Do this or else".

Two key exceptions exist, two aspects of the Internet that are fundamentally centralized: the distribution of IP addresses and the distribution of domain names.

Although technical problems exist with IP addresses—shortages in many regions of the world—these did not raise much in the way of policy issues. It is the highly visible domain names that triggered concern. This was a time before search engines were highly effective at returning relevant results, and people relied heavily on domain names just to find sites.

When someone has to make a decision that affects the whole globe, such as who gets to use the name praxagora.com, it raises the pressing question of what institution is ultimately in charge. Even more difficult is how to represent the interests of a mostly unknowing and indifferent public against the interests of actors with narrow agendas. Distributed and representative governance—that's the dilemma of the Internet, and it first opened its fearsome jaws around domain names.

Confusion in domain names could muddy a company's brand—and by 1994, companies understood that they needed a clear and unblemished presence in the domain name system even more than they needed a commanding physical presence out in the real world.

Most eyes were on the top-level .com domain, because it didn't have restrictions (.mil was for the U.S. military, .gov for U.S. government agencies, and so on) and because commercial use of the Internet was on a growth pattern that

> would outstrip the term "exponential". Internet-savvy speculators would cheaply buy up domain names corresponding to major brands and wait for companies to beg for them. Tens of thousands of dollars were traded, and companies were angry. There was talk of exerting trademarks and other actions that would place large companies in the driver's seat over domain names.

My records show that my first article on the domain name controversy appeared in October 1997 in my weekly American Reporter column. My first position paper for CPSR came out in March 1998. Tensions between free speech advocates and corporations were coming to a head, and a major summit was organized to find a solution that all would accept.

Amazingly, the summit succeeded. Free speech activists acknowledged the problems faced by corporations, and the lawyers representing those corporations recognized that the domain name system should serve the public interest. I didn't attend the summit, but I created a well-received set of principles with fellow CPSR member Harry Hochheiser. I gave our paper the name "Domain Name Resolutions", which plays off of the technical aspects of domain names. (Turning a domain name into a server's Internet address is called "resolution".) The various sides of the debate engaged honestly, transparently, and positively. A solution fair to all was in our sights.

Years of questioning never revealed to me what forces intervened to wreck the DNS agreement or why. The sequence of events was as bizarre as it was relentless. Someone threw together, with no guiding precedent, a bureaucracy called the Internet Corporation for Assigned Names and Numbers, to which Jon Postel abruptly turned over control of IP addresses and domain names. He died at the tragically young age of 55 shortly thereafter, and lawyers took over the vacuum left by the engineers.

The consequences of ICANN's hasty formation, omitting consultation with those who understood Internet names and numbers, showed up instantly. The obscure creators of ICANN stocked its board with people deliberately chosen for their lack of knowledge about the technologies they were supposed to administer. I had the historical privilege to attend the board's first meeting, which took place publicly in Cambridge, Massachusetts. Activists who had spent years investigating DNS and advocating for sane policies challenged the ingenue board members, who were obviously out of their depths and couldn't handle the most basic issues, such as how to run meetings that the public could engage in. The organization got only more distant from its constituents as time went on.

I should admit that ICANN actually found some support in the established Internet community. Some respected members of the Internet Engineering Task Force, which creates the standards over which Internet traffic runs, came out in ICANN's favor. The noted commentator and investor Esther Dyson became its first chair. But I believe these stand-outs were in the minority among activists. In a few years, telecom policy would similarly divide Internet activists. Many would call for stringent restrictions on telecom companies' discretion over traffic shaping, but came up against opposition by important leaders like Internet developer Dave Farber.

It's noteworthy that ICANN futzed around endlessly with domain name and trademark policy, but never deigned to look into two real problems that could reasonably fall under its purview: the security of the DNS root servers, which are constantly under attack, and the starvation of Internet addresses, which called for a smooth and expeditious transition to a standard called IPv6.

Even though ICANN was clueless about their domain of operations, nobody was more expert than they at creating expensive bureaucracy. I do not remember who decided to break the simple job of registering our domain names into two parts. But ICANN became responsible for regulating two types of organizations, at least one of which was redundant. The first were the "registries", which were responsible for maintaining and sharing the mappings between domain names and addresses. The second were "registrars", which were responsible for charging an exorbitant fee to anybody who requested a domain name. I don't know of any other responsibility assigned to the "registrars", except those that they picked up over time were part of the superstructure plastered on top of DNS by ICANN.

Now and for some time after, I was writing articles about DNS and soon ICANN on an almost weekly basis. Because large corporations asserted their right to control DNS on the basis of trademark law, I learned a good deal about that corner of legislation.

ICANN pressed an even bigger issue on policy-makers from all corners: How to govern a virtual space? Nominally subordinate to an obscure sub-agency of the U.S. government, the National Telecommunications and Information Administration, ICANN effectively reported to no one. Everyone agreed that its historical relationship to the NTIA was arbitrary and should not continue, but the dilemma was how to grant some accountability to the public.

There are many facets to this conundrum, which can't be resolved in a manner as simple as John Perry Barlow suggested in his utopian Declaration of the Independence of Cyberspace. Currently ICANN is in the astonishing position of being one of the few organizations in the world that officially has no accountability to anyone (except a fragmented internal collection of stakeholder

representatives, structured as Byzantinely as everything else ICANN does). I reported heavily on the hot debates taking place around Internet governance.

Hardly anybody now remembers how close the DNS stakeholders in the mid 1990s came to creating a supple and well-run DNS system. Victory was snatched from us by obscure forces who have created the boondoggle that funds junkets for ICANN board members and creates uncertainty for all.

The hegemony is eminently visible in the unconscionable prices charged for domain names, although questionable policies extend deep into the system. Technically, a domain name consists of a row in a database with information about the owner, and an entry in a table in the memory of a server that resolves the domain name. That all costs a tiny fraction of a cent, but domain names go for $15 or more. Monopoly charging supports the kind of gargantuan, complex, arcane bureaucracy beloved by large corporations and their overcompensated attorneys. Companies that choose to take away a domain have several options for doing so, while individuals and non-profits are left defenseless. In line with classic strategies for maintaining control and creating conflicts that only the bureaucracy can manage, ICANN enforces artificial scarcity by limiting the creation of new top-level domain names such as .info.

Along with the other activists who had gathered around the domain name issue in the mid-1990s, I continued to follow ICANN's tortuous organizational proliferation and decision-making for about a decade. Some of the activists, more tenacious than I, hounded the organization much longer. Someone even set up an organization called ICANNWatch, whose last posting dates to 2010.

Toward the end of my long dance with domain names, I received a juicy perk: an all-expenses paid trip to Paris. The locus of this visit was one of the international conferences called regularly by the Organization for Economic Co-operation and Development (OECD) to bring together all the member nations from around the world and assess their progress on various tasks. The OECD, of course, is one of those international collections of policy wonks who have an uncertain role but can spend enormous amounts of money on it. Many goals of the OECD are certainly laudable. Their Millenium Development Goals include such crucial initiatives as health, education, and environmental sustainability.

A group of Internet activists persuaded OECD leadership in the late 1990s to host an additional conference day devoted to Internet policy issues. Two of the organizers were Meryem Marzouki, a computer expert and political activist in Paris, and Marc

Rotenberg, head at the time of the Electronic Privacy Information Center (EPIC). They knew me from various policy battles, and I had met Marzouki at a Computers, Freedom, and Privacy conference, so they thought of me when they needed a speaker to cover DNS. They gave me eight minutes on the schedule, a cameo role for which they offered to pay flights, hotel, and meals for three days. That gives you a sense of the cash flow at OECD.

The Paris event was a great way to connect with people from as far away as South Africa with whom I had worked online. I wrote my talk, about the importance of access by small organizations to domain names, in an English style that I hoped would be easy to translate into French. I actually overran my eight-minute allocation by three minutes, but the talk was well received. And it was just my second day in Paris, after which I was free to enjoy the city. That's where I made my only mistake.

Having never been to a meeting of a major international policy organization (except a few minutes in the United Nations General Assembly as an elementary school student), I felt curious about the OECD. As an invited guest, here was my great opportunity to sit in on their sessions and see democracy in action. I set aside my last full day in Paris for this education in policy-making.

Naturally, the whole event was a bore. Bleary-eyed representatives from various countries intoned their progress toward various pre-set goals. I was watching it all on video in a windowless conference room. Had the organizers talked to us Internet activists, they could have done it all by email and saved their funders tens of thousands of dollars. Thus I discovered the frustration that I'm sure many people feel toward bodies such as the European Union. It's hard to get hold of a sense of citizenship in the face of such bureaucracy; it's easier to see a bunch of mediocre talents living high off the hog at the public's expense.

I finished my invitation to the OECD in their cafeteria, scarfing a lunch that couldn't boast of any camaraderie with the cuisine for which Paris is celebrated. I then headed a few blocks west for a visit to the glorious Marmottan Museum, my sole tourist indulgence before flying home.

Government takes to the Internet (1992)

This chapter has shown several of the critical policy issues where computer experts spoke up and made an impact on public life. Some stories featured CPSR, some involved other institutions. I'll finish with one small intervention we made in my local chapter of CPSR, early in my activism—an intervention with portent for a massive wave of change to come.

CPSR was willing to take its message anywhere. Our Boston chapter was one of the strongest, and in 1992 we conceived of lobbying our representatives at the Massachusetts State House to go onto the Internet.

Although we're one of the most progressive states in the country, the liberal values of Massachusetts legislators do not inspire any particular transparency and sense of public participation within their own deliberations. The institution is extremely hierarchical. For a while, an annual ritual brought the Governor, the Speaker of the House, and the Senate President together to secrete themselves in a room for a week or so and emerge with a budget at the end. They often missed their legal deadline for a budget, but the rest of the legislature and everyone else in the state just had to wait with bated breath for the termination of their conclave. When we talked to representatives about putting proposed bills on the Internet so that the public could watch the law being made, one representative said that sponsors liked to release bills on paper so they could keep all copies in their office and see which representatives came by to ask for a copy.

Four of us went to the State House one day and managed to hook up a terminal through their phone. It turned out that even their phone jacks were some strange outdated design, so that large infrastructure changes would be required to get their computers on the Internet. But we did establish a network connection. Sitting at the terminal, I took several representatives on a tour of the Internet. I showed a few basic functions such as mail, and ended with a flourish by demonstrating the most advanced tool of the time: gopher. You can look up this software if you haven't heard of it. It was an early implementation of hypertext, a pre-Web way of connecting information from various sites through a hierarchy of menus, all of course in text. The representatives were impressed.

We also met the IT staff of the State House, who were also enthusiastic. Eventually, this vision became a reality. The Massachusetts State House web site contains detailed information on bills, hearings, and votes.

Some 25 years later, a vast movement would arise in countries around the world for online government. Advocates within and outside government would call not just for putting governments and their staff on the Internet, but for using the Internet to draw the public deeper into policy-making than ever before. O'Reilly would latch on to that movement.

☞ *Juggling on a ship that pitches and rolls: Sponsors*

Juggling on a ship that pitches and rolls: Sponsors

Contents of this chapter

Judging a book by its cover: Authorship and attribution (2019—2020)

Some directives would seem to stultify the creative impulse, and yet end up spurring artists to their best work. For instance, when a resoundingly forgettable composer named Diabelli asked Beethoven to compose a variation on a silly little theme of no musical distinction, Beethoven wrote a set of 33 such variations, one of his finest compositions. It's interesting that this virtuoso achievement came toward the end of Beethoven's career (in addition to which he was deaf—but the music's inventiveness mocks that hurdle). Two of my final projects at O'Reilly also turned dull lead into gold.

The first project concerned programming languages invented during the 2010 decade. Programming languages have proliferated like fruit flies ever since the first experiments in providing a logical view of computing on top of machine language. Recent years have accelerated the urge to do new things with syntax and structure. So eight or ten languages of interest appeared during the 2010 decade, and managers settled on six for me to document.

The manager's outline asked simply for basic facts about Language One, followed by basic facts about Language Two, and so on. I immediately discarded this uncreative approach, as I could not imagine our readers wanting such a document. A reader could derive the key facts about any language from a glance at its web site or by reading a blog posting written by a curious programmer who had spent some time exercising the language's features.

The true opportunity provided by this project was to look deeper, ask why these new languages entered the scene, and draw general conclusions about the evolution of programming languages. Each generation of programmers faces pressures and requirements that make them impatient with current languages and drive them to seek something new. I realized that I would have a winner if I could identify little-discussed trends that united various innovations over the previous decade: the six new languages, upgrades to sturdy old languages of the past, and some experimental languages that managers did not ask me to include because they were not yet popular.

My report traced new directions in languages to a combination of needs: the ubiquitous use of multiple cores, changes in the Web, virtual computing and cloud computing, an evolving understanding of where most programming errors crop up, and a couple other big computing shifts. The first half of my 30-page report explained how these issues affected general trends in computing. The second half then furnished a page or two about each language. I didn't bother with a superficial overview of features, but instead fit the language into my framework of trends. Although this required a lot of broad characterizations, my luck held, for the reviewers approved my analysis.

With the draft ready to go to production, a manager informed me that they had decided not to put my name on the document. Instead, it would bear the byline "O'Reilly Editorial Team." I wasn't bothered by having my work reattributed this way. On the contrary: I found it supremely flattering that the editors would trust my research and analysis enough to rest the reputation of the whole team on this document. After all, a 30-page report about something as complicated as the history of programming is forced to trade in lots of generalizations, which leave the report vulnerable. The gamble paid off in this case.

However, I did maintain enough instinct for self-promotion to sign the document in a sort of textual steganography that I had learned from close readings of Shakespeare and the Bible. I knew that the authorship of anonymous texts could be rediscovered by comparing the documents in question with seemingly unrelated writings. And therefore, I went to another recent report that bore my name, plus an older article about programming that I had published on the Web, and extracted relevant sentences that I buried in the new report on languages. If I ever feel a need to demonstrate my authorship, I will pull out the articles and point to the

intentional plagiarism. The sentences I copied are obscure enough that no one but I would have been likely to be interested in repurposing them.

Another call to authorship, which mandated some fast knowledge cramming as well as creativity, started as a typical project for this phase of my career: An O'Reilly manager wrote that they needed a report on data engineering almost impossibly fast, because someone else had been assigned to write it and failed to come through. When I found out who the original author was, I snorted, because he had done exactly the same thing less than a year before: Sign up for some project, do a few interviews, and then peter out (although the other time he at least produced a weak draft that I could build on). I told the manager, "Next time, come to me first." My appeal was ironic in retrospect, because this was to be my last project at O'Reilly.

I had only a vague notion of "data engineering", which had appeared perhaps a decade before, but a quick web search of the term turned up a number of familiar tools I had documented before. So I checked back in with the manager to reassure her I could write the report. This report was being produced at the explicit request of a sponsoring company, so I interviewed a few of their staff, who listed some tools and topics they wanted to cover, but offered no guidance except an outline that turned out to be incomplete.

>>> What is data engineering? Let's put it in historical context.

> Major changes in technology produce new occupations. For instance, in the 1930s, thousands of cart operators who were shuffling goods around town by directing teams of horses had to learn to drive trucks. That's why the truck drivers' union is called the Teamsters, even though I expect that very few members know how to rein in a horse.
>
> Similarly, the modern "big data" era brings new tools, processes, and responsibilities to the old task of obtaining and maintaining data. This shift in the 2010 decade led companies to define the new job category of data engineer. Most data engineers are software engineers who update their skills for managing data. A more difficult transition faces database administrators, the people who maintained data the old way.

I quickly decided that this report should fall into a series of descriptions moving from a high level to successively lower ones, providing a hierarchy of windows onto different aspects of the topic. I'd start with a description of data today and the reason for the role of data engineer, then proceed through lower and lower levels to the virtual cabinet of tools available to do the job.

Although the reader would encounter topics from the top down, going from generalizations to details, my research took the opposite direction. I determined what each popular tool offered, fit these features into the role of the data engineer, and built up higher views of skills and responsibilities for that role. The popular data science tools were copiously documented through books and online sites, so I thought there was plenty of raw material to go on. But along the way I made a startling discovery: Although the tools were used by data engineers as well as data scientists, the documentation was addressed entirely to data scientists. I had to guess what each tool would offer to a data engineer. I thus sensed that I was producing a document of real value: the first general guide for an audience that had not been taken into consideration before.

This report, the last work I ever wrote for O'Reilly, followed the exact philosophy I used on my very first manual at my very first tech writing job almost 40 years earlier. I've never stuck to the technical information I was asked to cover, but scanned the field broadly to decide what would benefit my audience.

Although the sponsor's outline had elements in common with mine, it was not appropriate for the learning process I envisioned, and was inappropriate in other ways as well. For instance, I realized during my research that part of the data engineer's job was to provision resources—after all, how could you store data or tools without computers to store and run them? But the sponsor didn't emphasize resource management in its marketing, so it had forgotten to mention that key task in the outline.

As usually happened with these sponsored reports, the tight deadline applied only to me. The sponsor felt no urgency to respond to questions or review material. In fact, after doing all the writing myself, I went out and found half a dozen tech reviewers from various other projects. I did this partly to get a variety of viewpoints, partly because I wanted reviews from experts I knew and trusted, and partly because I correctly guessed from previous behavior that the sponsor would not provide a tech review.

The sponsor's outline, however, rose zombie-like later and created some problems. When the manager on the sponsor's side finally took a look at my work, and admitted that he liked my approach, he expressed worries over my failing to followed the original outline. It turned out that his company had planned on receiving a modular document that they could break into pieces and roll out as blog postings over a period of months. This plan had never been conveyed to O'Reilly. Luckily for me and my integrated concept, our manager at O'Reilly told them they were contractually forbidden from breaking up the document that way. I suppose that O'Reilly's branding strategy called for keeping the document together in a recognizable format that advertised our imprint.

Still, the sponsor's representative wouldn't let go of the modular outline, and asked us to restructure the report to conform to it. I should point out here that there were some things in the outline I couldn't do—such as case studies—because the sponsor was supposed to provide them and didn't, as well as other things I couldn't do because they demanded a history of work in data engineering, not just the facts that could be gleaned from research. The representative was quite accommodating, which was a relief, and seemed unconcerned that some topics had to be left out. So it's beyond me why he wanted to force my carefully paced document into an arid format totally unsuited to the educational thrust of the report.

I faithfully pursued the task, but ended up making superficial changes such as renaming sections with names from the original outline. And I felt I could breathe again when the sponsor accepted my work without further complaint about the outline. They satisfied their marketing goals by adding a foreword, and approved my draft to go ahead to publication.

This story had a happy ending. The rest of the chapter contains other stories about sponsors, some with happier endings than others. But for a moment I will turn aside for some musings on ghostwriting.

During my routine review of changes made by the series editor to my data engineering report, I noticed an odd change to a standard disclaimer I appended. After thanking the reviewers, I had written, "Any errors should be attributed to the author alone." That was a typical way to accept responsibility found in almost any factual report. But the editor changed "author" to the plural. She also made another addition implying that the sponsor had an author on the report.

I took this all in stride, knowing from past experience that someone else might be added, until further inquiry revealed that I would not be acknowledged in the report at all. Instead, it would be attributed to two people whom I had interviewed near the start but who had done nothing else since then. Still, I did not complain. I really don't mind when O'Reilly attributes my document to some employee of the sponsoring company, or adds their name to mine. I just like to see the money coming in. I'm doing work for hire, and part of the deal is to go along with management wishes on things such as attribution.

But I feel embarrassed on behalf of the sham authors. Did they ask to put their names on something they didn't write? Or were they simply chess pieces in some corporate marketing plan, as I was?

I sometimes imagine one of these putative authors speaking at a conference or before a team of customers, and facing a sudden challenge from someone: "I don't agree with this claim you made in your O'Reilly report..." Would the celebrity non-author

then admit publicly that they never had anything to do with the report? Try to talk around the challenge? Meet it head-on? Beg off by saying the report was written some time ago and the non-author doesn't remember the details?

Is it simply thievery to speak about something as if you invented the idea, when in fact you took the words from somebody else? Jewish tradition, which deals explicitly with this situation, treats such deceptive attribution as equivalent to a heinous crime. I can be more gentle and imagine the sponsor reasoning thus: "Our leading staff are experts in their field. The report we're sponsoring is what they would have written, had they the time and writing talent to do so." Yes, I could see some legitimacy to applying that logic to quotidian reports, following predictable paths, that technical companies routinely produce. The logic is less defensible when examining my own work, which delights in unexpected tangents, striking examples, and metaphorical comparisons.

Now my inquiry into ghostwriting enters into some thorny philosophical questions. What is more relevant: the concrete falsehood of claiming authorship over another person's work, or the deeper truth that the person whose name on the cover is an expert in that area? Does the sponsor really extract more business out of that message, conveyed as it is through an inauthentic claim, and is the practical benefit justification for this whole intricate dance? Finally, does O'Reilly's complicity, and my own willing participation, seem more permissible or less so, knowing that our revenue and my livelihood depend on that participation? I leave these questions on the table for future discussion.

Only in one situation did I demand recognition for my work, and it was as editor instead of writer. By the 2010 decade, I was rarely proposing books. But once I got the idea that we needed a book about security in the cloud. Although in the past we published books by some of the leading experts in computer security, our security offerings were currently were in something of a slump. Because of staff departures, no one was assigned to that crucial field for a while. So I took independent action on my idea. I found an author by talking to friends in the field, worked out the proposal, submitted it for approval, and arranged the schedule and contract. I also edited the book and dealt with the tech review. The project lay entirely between the author, tech reviewers, and me until the book went to production. This was the last project to proceed as I had operated during my golden era of the 1990s.

When production sent me their edits, someone else was listed as the developmental editor. When I pointed out the inaccuracy, I was informed of a new policy I had never heard about before: They liked to put the managing editor of each series on every book of the series, in case someone may decide to contact us and propose a new book.

I puzzled over this reason for denying me credit. I questioned: Would computer experts really think like this? "Hey, I'd like to write a book on some topic, so I'll find a book on a similar topic from O'Reilly, turn to the copyright page everybody ignores, check the fine print to find a name, and dig up their contact information." So I found a solution acceptable to all. I proposed that the managing editor of the series be listed along with me as developmental editors, and that compromise was accepted.

On the data engineering project, too, I decided to take action and ask for my work to be properly attributed. I did so because I suddenly realized that proper attribution would promote O'Reilly's interests. I reached this conclusion when the project manager on the sponsor's side mentioned, in a conversation with me and two project managers at O'Reilly, that he had read a report I wrote on streaming data eight months before. He went on to praise the report highly, talking about how I had conveyed concepts at just the right level and in the right order. I got the impression that seeing my successful work made him and his team more confident in the quality of what I was currently doing for them.

It is ironic that this manager would demonstrate the value of having my name on a good report, and then try to take my name off of the one I was doing for his company. If this were to happen often enough, potential sponsors would no longer get to see that I was the author of unusually interesting content. This would prevent me from getting more compliments, but that's a trivial point. More importantly, it would prevent O'Reilly from using my name as an incentive to sign up sponsors. Wouldn't it be nice if, during negotiations over a contract, an O'Reilly manager would say, "Andy Oram will be the author," and the sponsor would say, "Ah, yes, we have seen Andy's work and feel confident of him doing the job well"?

So I called Colleen Lobner, who managed much of the logistics for sponsored content at O'Reilly, as well as my direct supervisor, Rachel Roumeliotis. Both of them understood my points and agreed with me in principle. They promised to take my concern into account on future sponsorships. And when the data engineering report actually came out, it bore my name as sole author. I am quite proud of this contribution: the first published explanation of the data engineer's responsibilities and the support provided to them by modern tools and processes.

I approved the copy-editor's final version of the data engineering report the week before the layoff, so this report represents a fitting capstone to my 28-year career at O'Reilly. Managers would no longer have the opportunity to refer to that report, or any other, in order to promote me as author.

November 2020: There is already an update to the sad would-be epitaph to my O'Reilly reports in the previous paragraph. This month I was brought in as a free-lancer to do more reports for them, a new relationship with the company that I welcomed.

The data engineering report typifies the process I went through repeatedly during my final years at O'Reilly: sponsored content. This was content that looked just like everything else we wrote (a web article, a report, perhaps a whole book) but received some payment from an external company.

What did sponsorship accomplish for the sponsor? I have to admit that I never saw a contract with a sponsor. (It was suggested early on that I could help negotiate contracts, but ultimately I was never brought in until the contract was signed. This could prove awkward, as we'll see, when it turned out that the topic or outline provided to me was inappropriate for the subject matter.) My impression was that we built a successful program around a pretty small payback. A sponsoring company might just get a chance to put their name on the cover of a report and offer it for free on their web site for a limited time—say, three or six months. They could strike gold during this time by collecting email addresses from visitors as a prerequisite to downloading the report.

The sponsor could also benefit by explaining key concepts they wanted potential customers to know in the report. These concepts might or might not pertain directly to their own services, often it consisted of more general coverage on some topic like the data engineering report I discussed before. Some companies more or less outsourced to O'Reilly the task of providing detailed instructions for tools hosted by those companies. I suppose there was also a prestige value in having their names associated with the content.

Whatever companies saw in sponsored content, it proved highly attractive. Our program grew. We hired even more people to focus on the program after Hendrickson left the company. In this chapter, we'll look at some surprising things sponsors asked for in exchange for their sponsorship, and the impacts that had on me and on O'Reilly.

An expensive book signing (2017)

O'Reilly always included a statement about its editorial independence on sponsored content. During contract negotiation, managers explained to sponsors that we needed to make the final choices about what to say in order to maintain the readers' trust, which would in turn benefit the sponsor who wanted people to read our work. No marketing!

The editors were pretty strict about cutting material that gave too rosy a view of sponsors' products or services. Still, as the previous story showed, sponsors can exert a strong influence. We often changed the title or content to satisfy a sponsor. As the previous section showed, I was willing to mess with a strong structure in response to arbitrary requests. There was a general tendency to be helpful to sponsors.

I fell right in with this helpful atmosphere. But once it got me into trouble when the oxygen was suddenly removed.

One day in 2017, I heard from O'Reilly's conference staff about a sponsor request. A few months before, I had supplied this sponsor with one of the many reports about data analysis using artificial intelligence I was being asked to produce during the late 2010 decade. The O'Reilly staffer told me the sponsor would like me to come to a Strata conference being held imminently in London, to sign printed copies of my report. Author signings draw visitors to a vendor's booth, and many companies are willing to invest a considerable amount of money to bring an author in. Additionally, this company wanted a short posting about algorithms for their blog.

For Strata, the conference staff person explained that the sponsor was willing to pay my flight, meals, and several days at a hotel in London. Being in a hurry to reserve the flight, room, and conference pass, I polished off all these tasks after a flurry of email with the conference staff and with a marketing person from the sponsor. Finally, I let my boss Rachel Roumeliotis know that we had made these arrangements.

Reading along, have you been able to guess what I did wrong? I should have brought Roumeliotis in at the start, before reserving the flight and other resources. I had stepped on a land mine. Within hours I was on the phone not only with my Roumeliotis but with Mike Hendrickson, who ran the sponsored content program. They told me that the scandal had risen to the level of C-suite, with Laura Baldwin as irate as the rest. As you will see later in this memoir, I considered all three of these people central advocates who made my continued employment at O'Reilly possible.

Here's what I gathered from the phone call. The team responsible for sponsored content wants to do all negotiations with every sponsor. If the sponsor wants something extra—such as my attendance at their booth, or a fresh article—they must go to Hendrickson and his team and work out O'Reilly's share of the compensation. The sponsor had gone around Hendrickson's team and won valuable services from me personally without paying O'Reilly (aside from my conference pass, I suppose).

I wondered why the sponsor was ignorant of this rule. Who, among the ranks of personnel who report along sometimes skewed lines of hierarchy, and who are probably too busy to check every box and scrutinize every clause, would be responsible for knowing that they might be violating their contract? The sponsor was not a huge conglomerate, but perhaps someone should have consulted a lawyer before asking me for extra work. Perhaps the lawyer would know how O'Reilly management would treat the request, and perhaps not. And perhaps the sponsor assumed that staff at O'Reilly would enforce expected behavior.

And that pondering took me back to the mode of decision-making at O'Reilly. Why were the conference staff as unaware of proper protocols as the sponsor was? And who else had been involved during my hasty negotiations? I felt shaken up enough to do a little personal backtracing, and uncovered email that cc'd a person on Hendrickson's sponsored content team. Should this junior person have recognized that we were violating protocols? Was she responsible for keeping on top of contract details and recognizing all their implications? I didn't blame her, because the expectation that managers were fulminating about on the phone with me was not explicit.

One of my relatives sympathetically suggested a psychoanalytical explanation to me, opining that O'Reilly managers were envious of my success and didn't like to see my work promoted without getting some of the credit. While I wouldn't dismiss her slant on the controversy, I can well see that O'Reilly had a legitimate beef, because they perhaps could have wormed a better deal out of the sponsor than I did. I was thrilled to get a free trip to London, and the prospect of the extra blog posting stroked my eternal pull toward self-expression. By some calculations at a high corporate level, the sponsoring company might have been getting a lot on the cheap.

By the time they talked to me, O'Reilly management had already settled on what they felt was an appropriate response to the lack of communication of which I and the sponsor were guilty. They would pay nothing toward my trip to London, and would require me to call it vacation time. Furthermore, they would do nothing to promote my appearance there or the article I would provide to the sponsor's blog. And they followed through: No O'Reilly outlet put out so much as a tweet on social media about these things.

As for me, I proceeded along my merry way. I looked forward to publishing an article on a topic of great concern to me, to showing off my report to readers at the conference, and to getting into London for a lark. But I did feel worry and embarrassment—not for myself, but for my company. It is well and good to notify a sponsor that we expected different behavior—but what would O'Reilly gain by disregarding the sponsors' interests? Couldn't a sponsor be repelled by this behavior and cut back on future

support? Why couldn't O'Reilly managers show some tolerance for error, offering to join the sponsor to work on this final leg of the project? Perhaps O'Reilly could have met with the sponsor to convey their feeling that the separate agreement worked out with me was not fair to O'Reilly, and end up cutting a deal by which O'Reilly provided some support and publicity in exchange for a modest payment. But none of this was in the cards now.

From that point on, as another relative put, I made lemons into lemonade. This is how.

The blog posting was quick work. Handed the broad request to write about artificial intelligence (the topic of my original report for the sponsor), I decided to cover the pressing issue of bias, which you remember from the last chapter had been the topic of a summit I had attended a few months earlier. The sponsor was surprised that I had picked a topic with political muscle, but was so impressed by the density of references backing up my assertions, from both the popular press and academic papers, that they gladly published my posting.

The sponsor consolidated my commitments at the conference into a single evening, but still agreed to pay for several hotel nights. The hotel they chose was at the eastern edge of London near the yawning expanse of the conference space, but I reserved one extra night at a hotel in West London near Kensington Gardens, where I had stayed with Judy about a year earlier.

By a lovely bit of luck, this earlier pleasure trip we had scheduled to England coincided with a request from my manager to write an article on the tech scene in London. The purpose of commissioning the report from me was to interest Britishers and others in our London conferences, such as Strata. I seized the opportunity to schedule a meeting with some tech leaders during the vacation. I also took a number of photos that I worked into the report, along with quaint plays on words and other in-jokes that held meaning only for people who had spent some time in the capital. I don't believe that this report had an impact on conference attendance, but it's a nice overview that was well appreciated by the people I interviewed.

Making good use now of another trip provided by the sponsor, I resolved to be at the conference site not only the evening on which my commitments fell, but for the entire day. Thus, while having plenty of time to indulge my love of theater and museum-going, I could also derive whatever professional benefits came from attending the conference. I was genuinely curious about Strata, which I had attended just once in the U.S. I had never been at a foreign O'Reilly conference before.

Learning that London conference attendees dress more conservatively than the computer geeks who frequented U.S. conferences, I even packed my suit jacket into an old garment bag instead of bringing my usual suitcase.

The opprobrium directed at me by management on our phone call did not trickle down to other parts of O'Reilly. The conference staff was excited to hear of my presence and printed up not only the report I wrote for the sponsor, but the report on the London technology scene I had finished the previous year. I arranged to spend some time in the O'Reilly booth signing the report for attendees. I met several old colleagues from O'Reilly on my day at the conference, and with no knowledge of the controversy around my attendance, they all expressed pleasure at seeing me.

Hendrickson arrived, surprised at finding me hanging around the O'Reilly booth. He knew that O'Reilly was not paying me a cent to be there, and asked me why I bothered to spend any extra time at the conference. I explained that I was interested in the conference and felt it a great opportunity to attend.

The book signing at the sponsor's booth went superbly. I formed bonds with staff from the sponsor and devoted time to each person who picked up a report, penning a unique dedication for each. One attendee was a data scientist for a Russian bank whose name sounded familiar to me. I was tempted to tell him, "My country just put sanctions on your company," but instead I just wrote his name and a nice statement in the report. The marketing representative, who was a fascinating person herself—she had been in professional theater before getting into industry—offered to end the book signing early, but I was having a good time and suggested that since we had come so far, we might as well stick it out to the end of the evening. We then socialized at a reception in a nearby pub.

Having fulfilled all my responsibilities and sampled what Strata in London had to offer, I spent the next day at the National Gallery, where I satisfied a long-held yearning to see their extensive collection of Turner paintings. And I finished up the trip by having a meal with cousins. I gave them copies of the two reports from the conference, thanking them for some information they had provided to my London report.

But I was not totally finished milking my trip to London. One of the sessions I attended at Strata covered the recently passed European directive on privacy, the 2016 General Data Protection Regulation. A free software publication I write for occasionally was interested in what I learned at the session and paid me for an article I published with them upon my return home. This sum exceeded the expense of my extra hotel night and other costs of the trip that had not been covered by the sponsor.

The contested survey (mid 2010s)

The previous story recalled a time I angered O'Reilly management. This story will cover the best thing I ever did for O'Reilly: killing off a project.

This eccentric achievement called for a combination of my editing skills with analytics. Such combinations of skills are commonly required in a small company, where at unexpected moments one has to pick up tasks outside of one's job description without even stopping to think about it. Those of us hired to be editors when the company was called O'Reilly & Associates always did much more than writing or editing. So it was as amateur data analyst as well as editor that I helped the company avoid a publishing scandal.

The story starts with a keynote speech at a Strata conference I didn't attend. I'm sure that Cloudera, one of the first companies to offer machine learning in the cloud and now a major player in cloud computing, paid a good deal of sponsorship money to land a spot on the plenaries of an O'Reilly data conference. Here, on the stage before the entire assembled attendees of the conference, one of their top managers rang out his conviction that the computer field was undergoing a seismic change—with Cloudera at the center. This kind of insistence is common to many managers, which I can understand because they have thrown years of their lives into building a company and want to believe not only in their material success but in their historic importance.

Although industry trends suggested to more objective observers that machine learning in the cloud would spread gradually and in tandem with older forms of data analysis, the Cloudera manager announced us on the cusp of an era where older offerings would shrivel up and vanish. And he overstepped good sense by picking out a particular representative of the obsolete past, a company I will refer to simply as BigCompany because of their later schemes. The BigCompany's of the world are on their way out, thundered the Cloudera manager, declaring that their offerings would soon be irrelevant.

Perhaps the Cloudera manager did not know that leading executives of BigCompany were sitting in the audience at that moment. I suspect he did, because BigCompany itself was a major sponsor of the data conference. In any case, they were understandably outraged and sought out the manager of O'Reilly's sponsorship program to complain.

This manager, as you know now, was Mike Hendrickson. Suave, square-jawed, and athletic, with a hint of unresolved brashness, Hendrickson seemed to me superbly cut out to sell our services to large corporations—just the skill O'Reilly needed as it evolved into what the agile-tongued business analysts call a "B-to-B" strategy (for "business-to-business"). Sponsored content was

part of our strategic shift away from selling books as individual units, which is "B-to-C" ("business-to-consumer"). Although Hendrickson had managed several editors, when he first came to O'Reilly, including me, he really thrived in sponsored content.

The way Hendrickson handled the BigCompany complaint demonstrates his salesmanship. He converted their anger into yet another sponsorship opportunity. "You clearly have an important message to convey about your offerings," he told them. "Let us help you get that message out." And they took the bait.

I had no role in drawing up the agreement between BigCompany and O'Reilly, but was brought in after they settled on a plan. BigCompany went in big. They proposed not just one, but three reports about their offerings. The first report would cover traditional tools, the second would cover the new ones that Cloudera had championed (and that BigCompany had now added to their service as well), and the third would tie the old and new together.

That was not all, however. BigCompany also designed a survey about people's use of analytics in the cloud. This survey would be offered on their site as well as O'Reilly's, and would ask numerous questions about the customers' background and the background of their companies: how long their company had been in the field, its size, its geographic reach, and so on. Correlating the data would presumably turn up trends that were both statistically significant and useful for planning. BigCompany seemed confident that the trends would underscore the value of their own motley offering.

With this scaffolding in place, I was brought in to write the three reports as well as a web posting explaining the results of the survey. I examined BigCompany's offerings as well as other available material on the three topics, and soon encountered some disconcerting limitations.

The first report concerned familiar topics in databases and data warehouses that had exhaustively covered over a 30-year period. BigCompany was essentially a cloud provider before the term "cloud" came to be applied to third-party computing services. BigCompany did not advance the field in any way. They simply relieved computer administrators of some of the fuss and trouble of running familiar tools, no doubt leading to cost savings. This was a nice, if well-trod, marketing story, but devoid of technical interest.

Turning now to the second report, I checked BigCompany's offerings again and saw nothing to make a reader's eyes widen. The free software community had been very fertile in the areas of big data and machine learning, producing well-known tools such as

Hadoop that had taken the computing world by storm. All the cloud vendors added the tools to their offerings. Any enhancements added by the commercial vendors went back into the free software versions and were duly documented.

Some cloud vendors did create their own versions of the popular tools. Sometimes, as in the case of Google, the proprietary versions preceded the free and open source ones. I compare the relationship between free and proprietary offerings to travel in a foreign company. When you enter a grocery store in the new land, you will see popular brand names from your own country—the grocery equivalent of free software tools in the cloud. You will also see slightly different versions of the same foods under local brand names: These are like the proprietary tools. BigCompany did not even create knock-off proprietary versions, so I had nothing to say about the new analytic services they boasted about.

Thus, I persuaded O'Reilly management that the first two proposed reports would be redundant, and they brought BigCompany to this realization by promising them the single report that combined the two types of tools. Here, again, Big Company had done nothing innovative. By now, I was coming around to appreciating the accusation leveled by the Cloudera manager and suspected that BigCompany did not, in fact, have the message that Hendrickson encouraged them to offer. Still, I found a way to write up about 25 pages describing how to link together the old data warehouses with the new analytics. The report was serviceable and usable, although I did not rank it one of my important contributions to computing. O'Reilly sent it to BigCompany for their opinion.

BigCompany didn't want to publish the report. Their managers had to sheepishly admit that what they really needed was marketing, which O'Reilly would not provide. My technical description, although accurate, was not meeting their needs. We threw my work into the virtual trash.

But one BigCompany project yet stood—do you remember it? Yes, the survey. We had promoted the questions and received a lot of answers. A data scientist on the O'Reilly staff performed some routine correlations and came up with about 35 graphs showing all kinds of relationships: the relationship between company size and success, between geographic location and the use of machine learning tools, and so on. We had a plethora of results to sift through, and there was potentially something to learn from each of the 35 slides, whether or not it showed a statistically significant relationship.

Now I was cooking. I detailed everything that might be considered counterintuitive or surprising in the graphs. Some of the questions produced no information of value, and I often wished that I had been granted a chance to review the questions before the

survey started. Nevertheless, I derived many insights that I grouped into sections of an article and produced a strong draft.

But once again, this piece had to be approved by BigCompany. And once again they were disappointed. They had been hoping for results that directly justified their business model. And my summarized insights, interesting as they were, didn't match up with the message they were telling customers. So they threw out this second article as well, and told O'Reilly they would bring their own author in to summarize the survey results and produce the article they wanted.

This is where a critical choice made by O'Reilly management determined the fate of the project—and our own reputation. They didn't simply publish the BigCompany draft, but showed it to me first. Whether they wanted simply my editorial eye or something deeper, I don't know. Curious to see how BigCompany could improve on what I had done, I dug in and scrutinized their article.

I had to admit that I was impressed. The author was masterful at explaining both technical details and business implications. His explanation of the results from a BigCompany point of view was persuasive, strongly stated, and eminently readable. There was only one problem: It was completely fabricated. The author blithely ignored the clear evidence turned up by the data and wrote up BigCompany's desired outcomes, claiming support that didn't exist from the data.

The risks posed by this piece shook me deeply. If O'Reilly had gone ahead with BigCompany's plans and put the article on our web site, it would ruin our reputation. Someone would probably get their hands on the original data and expose the lies. Even without access to the survey data, people could report from the field that their observations contradicted ours, and get into endless incriminations that would hurt us more than it hurt BigCompany. I knew I had to squash this article.

But I also knew that O'Reilly managers wouldn't take me seriously if I just told them I disagreed with the BigCompany author. I had to give them hard evidence—the kind of fact-based background missing so much in the fake news that became prevalent at the end of the 2010 decade.

So I went back to the 35 graphs and meticulously found a graph for each claim in the BigCompany article. Going sentence by sentence through the article, I matched each claim with a graph and explained why our data disproved the claim. I created my own five-page comparison and sent it to O'Reilly managers. They then showed it to our data scientist—the one who had done the original graphs—and luckily, he backed me up. We turned down BigCompany's article.

At that point, I urged my managers to go live with my original article. It was accurate, I had invested a lot of time in it, and it had useful insights from which readers could benefit. But without sponsor approval, O'Reilly couldn't do that. My article was suppressed permanently. Instead, O'Reilly told BigCompany they could do whatever they wanted with the data, but just not to associate O'Reilly with anything they published. Thus ended a moment fraught with peril.

Writing for sponsors (mid 2010s)

You may be wondering how I persisted at O'Reilly during the final years when I made few acquisitions. Why wasn't I laid off all that time? I often wondered that myself. The company was by no means averse to layoffs. As with the conference team, they could close down entire underperforming divisions and send away the staff. That happened to the video company that they acquired on their way to developing a video strategy, and to a team conducting online training.

During times of unprecedented social and business churn, I think that I actually rode the front of an evolutionary wave—the peer-to-peer trend, discussed in the next chapter—and helped O'Reilly change its character. At other times I can only say that I had a lucky clinch with changes at the company, or even that I managed just to survive change. Writing and editing sponsored content was a matter of survival for me.

Mike Hendrickson, who had been my manager for a few years, moved into the role of signing up sponsors for this program. And he tapped me as an author and editor. This gave me a new lease on life—and a kind of influence that I usually lacked as editor: a chance to publish my own ideas. I wasn't kept busy all the time, but enough of the time to make me valuable. No longer responsible for networking with industry leaders and recommending books, I turned into a writing and editing machine. Managers would feed in project requirements and I would spit out results. When I did return to my old editorial role—signing an author or editing a book relegated to me—the project aroused nostalgia for the relationships I had formed and the sense of accomplishment I felt in those earlier years.

Standard business models that involve unit sales or subscriptions require an adept understanding of consumers and producers: our readers, our viewers, our authors. I have become comfortable over the years at balancing the views of my authors—and often maintaining balance among several authors on one text—as well as tech reviewers and sometimes the opinions of fellow O'Reilly staff. With sponsored content, the needs of the sponsor are tossed into this mix. How much harder can that be? Well, I soon

found that the pressures of sponsorship brought me just as much responsibility for O'Reilly's success and reputation as I had exercised in earlier phases of my career.

To fulfill writing requests promptly on any topic, I drew on my long experience with the history and use of computer technologies. I applied to each project the understanding I had developed through dozens of projects in programming languages, databases, security, analytics, and web development. This background permitted me to pick up every project offered me, and complete it quickly with insight. The new job left me plenty of free time to read about hot topics such as machine learning and keep pace with developments.

But my role was also restricted. I rarely got sent to the conferences I used to love, and I found less and less reason even to attend local hang-outs for techies where I used to network intensively. In my first decade at O'Reilly, the editor who worked on authors on one book would continue working with them on the next—that was part the editor-driven culture I have described earlier. But during the later years, other people in the company would routinely contact authors I had recruited and edited, then start new projects without even telling me.

Going further beneath the surface, my new role needed all the people skills and project planning savvy I had developed over the decades. These aspects of my work served both to make sponsors happy and to extract from them the information I needed.

The most amusing application of hard-earned skills earned years before came unexpectedly. A sponsor asked me to write an article on their proprietary technology, but the marketing people in charge of the project provided no technical information to start me off. They did dump piles of unrevealing marketing material on me. On projects like this, I routinely had to sift through the cruft to see whether there was anything worth saying—and rarely did I find anything. That was the case for this article as well.

Finally, after I pressed the marketing people one more time, they declared that all they could give me were two patent applications submitted by their technical staff. I don't know why they made such an offer, the absurdity of which must have been known to them. Patent applications rank notoriously among the most obscure and unreadable documents in modern life. I think the managers must have gotten tired of the project and wanted to get rid of it. I no doubt astonished them by enthusiastically requesting the patent applications. And in fact, thanks to the skills I had gained working on Peer to Patent and other projects as a policy activist, I extracted the technical nuggets from these applications that enabled me to write a comprehensible technical article.

March 2015, according to my list of publications, saw the release of my first piece of sponsored content. I didn't feel safe during the first few years of my transition, because nobody ever told me that I was making a transition. Managers must have clocked my output. They must have known that I was editing hardly any books and proposing even fewer. But no one said to me, "Andy, we know you can no longer perform the traditional roles of a senior editor, so we're going to assign you new responsibilities as a producer of sponsored content." For a year or more, I remained petrified that I'd be called on the carpet for failing at my responsibilities.

Occasionally I met my supervisor, Roumeliotis, and plied her for reassurance. I'd say openly, "I'm not recruiting many authors—is that OK?" And she always affirmed that what I was doing was what the company wanted. After a few such meetings, I finally relaxed. And the arrangement actually worked for a long time.

☞ *Thousands of hands to the tiller: Writing as empowerment*

Thousands of hands to the tiller: Writing as empowerment

Contents of this chapter

New media, new surprises: Blogging and public presence (2000s)

I'm slow to try new platforms, whether sites for self-promotion like Medium, or entire new types of outreach like Twitter. My attitude differs from many wannabe super connectors who go online and seem to be everywhere. I prefer to make the most of a few outlets, not to subdivide my time and attention into finer and finer shreds. I ended up on Facebook, LinkedIn, and Twitter. And I started late in the game, not feeling at first the need to send or receive wisdom in single-sentence packages. But an action by Tim O'Reilly, who attracted nearly two million followers at the time, persuaded me of social media's perhaps insidious power.

In 2009, both Tim and his company were heavily invested in a movement called Government 2.0 that appears later in this memoir. At that moment, I analyzed the structural risks and weaknesses of open government. I listed 19 problems that governments had to resolve to exploit data and engage the public. I prepared a draft that I reviewed with several people including Tim. For easier review, I put the draft on my web site but asked reviewers to keep the location secret.

Security through obscurity didn't work. Soon I was getting messages about the article from people I had never heard of. Luckily, the reviews were all positive. Asking around to uncover how the URL got out, I learned from Tim that he had put it in a tweet. My roster was now published, whether or not I felt ready.

I've been told that Tim was actually the one who invented the use of Twitter as a way to share news articles. Before he did so, it was a swamp of trivial observations about what people were having for breakfast or what their cats were playing with. Whether or not Tim played the historic role of turning social media into news feeds, his action of publicizing my article showed me how one could benefit professionally from Twitter. I joined up and used it a few times a week.

I was slow to get to blogging a decade earlier, too. When I heard about weblogs, I considered them a deplorable narcissism. I said anything I spent time reading should be thoroughly researched and carefully fashioned, preferably with review by a separate editor. Who would want to peruse someone's random jottings day after day?

In fact, I still feel that way about weblogs. Most postings I've seen are either cotton-candy ephemera that wouldn't interest anybody except the author's mother, or a gee-whiz expounding of ideas I've seen a dozen times in other places. True insight is rare. Although I track the blogs of a few leading thinkers, I notice that they tend to post only once every few months. Furthermore, those blogs all go dead after a couple years when the authors find something more productive to do.

The O'Reilly web staff persuaded me to start blogging. My records show that my first posting went up February 2000 (now gone forever, like most of my postings). The title, "Promising Personalized News", suggests that I was hip to an important trend very early. Two decades later, in response to the launch of a curated news site by Apple Computer, I published a piece on the same subject, probably with much more insight gained from years of observing the phenomenon. (Because O'Reilly no longer ran a blog, I put this article on LinkedIn.)

Blogging enhanced my career to an incalculable degree, even though I lacked an explicit motivation to post. Only a few months after that 2000 posting, I hit a gold mine with a posting that led directly to our engagement with emerging technologies,

described later in this chapter.

For years after that I was blogging at least weekly on one O'Reilly site or another. The team kept setting up different domain names with different types of blogging; one of our most popular sites for a while was XML.COM. Yes, believe it or not, we thought XML would be the future of the Web and of publishing.

Most of the content I wrote for XML.COM is still there as I record this memoir, although the site is moribund. I cannot say as much for most of several hundred blog postings I wrote. They would have complemented this memoir, offering interesting views on the development of computing and the Internet, but they almost all got wiped out during one or another of the transitions initiated by some team of web developers.

I remember only once that O'Reilly's web team made an effort to create an archive of old content. Most of the time, they understood their job to be just to reflect the company's latest model for public relations and outreach. They clearly had no idea that there was any value to preserving articles such as my 2000 posting with such an august impact on O'Reilly and the computer industry—the article we credited with launching the company's focus on emerging technology and "alpha" programmers; the article that ultimately led to our new mission statement: "to change the world by sharing the knowledge of innovators." It's as if only the future matters, not the past.

I must share some of their guilt. Writing frequently and hastily, I didn't bother during the early years to keep copies of most of my pieces. I might have undervalued my own writing because it was just a blog. Or until I started to notice the disappearance of my work, maybe I couldn't fathom that a company devoted to high-quality content would casually discard its own historical treasure.

December 2020: After some intensive web searching, I discovered a broken link to the historic article just described, which permitted me to retrieve the text from the Internet Archive, also known as the Wayback Machine. I have now restored the article to my personal web site.

At some point, management decided to shut down the weblog. The manager of the site told me they would concentrate on more substantial articles, giving authors time to do research and requiring them to go into more depth. I still wrote for them occasionally, when I had a topic such as algorithmic bias that deserved the research effort. Although I see good reasons for their shift in priorities—after all, thousands of other sites offer blogging opportunities—I regretted the loss of a place to post short pieces with

the attitude of, “Gee whiz, here’s something new and potentially important—let’s pay attention to it.” I think such postings could generate in readers an anticipatory thrill of discovery, even a sense of wonder, that would bring them back to our site.

The company’s more focused approach to the web site paralleled a general narrowing of the topics on which we published. To get the biggest bang for their buck, management sought groups of related products we could bundle—for instance, content on machine learning. The new policy disparaged individual books in new areas; every book had to fit a strategy chosen by management.

Even before blogging, writing for a number of different web sites (and often print media too) maintained a constant hold on me. As I explain elsewhere, I wrote a weekly column for three years about Internet policy, from 1997 through part of 2000. All in all, I have over one thousand articles under my byline.

The variety of topics I blog about is so broad that I can’t even summarize them. That tripped me up once during a juicy networking opportunity.

An O’Reilly convention I had attended was a wrap, and I took a flight to San Francisco for my usual activities following any West Coast conference: visiting friends, staying with relatives, and dropping in on O’Reilly’s Sebastopol office. It so happened that Tim O’Reilly was sitting on the plane a few seats away. As my wife pointed out later, Tim could easily have afforded a first-class seat, but his characteristic frugality put him in the economy class, so I had the opportunity to chat with him.

Tim told me that he had been invited to an important opening in San Francisco: the office of the Creative Commons, which was changing the way musicians and writers thought about collaboration and content distribution. He couldn’t attend, so he suggested I take his place. I had the honor there of meeting John Seely Brown, one of the most famous researchers in the intersection of computing and society—an area clearly very dear to me. When I mentioned that my work at O’Reilly included a lot of blogging, he naturally asked what topics I covered. I didn’t have an elevator speech ready. Despite my fervent hope to impress him, I couldn’t think of what to say, and ended up mumbling that I covered a lot of things.

How I can write about so many different fields and causes, I don’t really know. I have become a Jack of all trades just by being interested in the world and following my instincts toward whatever is interesting. That curiosity is what led me to investigate and identify peer-to-peer technologies even before that term was applied to them. It also prompted me to research health IT, an intuitive scouting out that led me to numerous conferences, hundreds of articles, and a White House summit.

Along the way I came to understand the meaning of the rather fusty word "autodidact", realizing that's what I am. In this trait I matched Leonard Woolf (a journalist and editor who married the more familiar Virginia). He said in his autobiography that he could edit a journal on any topic by spending a few months intensively reading about the field. This didn't make him an expert by any means, but it gave him enough background to guide the experts in producing material. I call this achievement the ability to develop a bullshit meter. You may not know the truth, but you can tell whether someone's assertions contradict each other, or violate basic principles of their field.

I've always learned more from my own reading and practice than from school. I did reasonably well in high school, but found myself less and less engaged in college. I always wanted to go off exploring angles that interested me, rather than what the instructor thought was important. That's why I never pursued an advanced degree after getting a rather flaccid B.A. in music.

This is nothing to be proud of. I highly respect people who can adapt to the discipline of academia—or who are so brilliant that they can meet its requirements while remaining focused on their personal vision. When I saw, at an MIT forum, a discussion among experts that produced dazzling insights, I realized the difference between being cross-disciplinary and my own learning, which is just undisciplined.

But are there advantages to flitting from subject to subject, drawn like an insect by the ever stronger attraction of nearby scents? Yes, I think so. My synapses are wired in such unusual, multifolded ways that I find surprising associations between unrelated ideas. I can carry principles into venues where they didn't appear before. In my roster of articles, I have never allowed myself simply to repeat what others have said before. Every piece offers readers something new to chew on.

What everything comes down to: Copyright (2000s)

The digital tsunami, which has forced so many changers in careers, politics, and how we live, also shakes apart the delicate balances that existed in copyright, patents, and trademarks—the main subtopics in intellectual property. Each of these subtopics in turn has leapt on the stage during my career.

Editing Van Lindberg's book *Intellectual Property and Open Source* came naturally to me, because I had been developing the kind of amateur knowledge I mentioned earlier for years in these areas. Lindberg, who later became director of the Python

Foundation, got a law degree as a young coder and maintained an appreciation for free software. He could easily fall into nerd-speech while talking about either law or coding, and was the perfect author for a book straddling those realms.

Our deliberations over the title were fraught with caution, more than the discussions I always hold with authors to choose the best title for their books. Every book needs a title that instantly draws attention while conveying the book's scope, even before the age of search engine optimization. The choice is easy enough if you're talking about one well-defined topic, and especially if your book is the only one about the topic: Just assign a single-word title, or one garnished with a modest gerund such as "Using". But most books have to fit into a pre-existing landscape of related titles and must somehow signal to readers that they'll derive some special benefit.

Our book faced its own unique hurdle because free software developers were a major audience, but the Free Software Foundation had declared "intellectual property" a word to avoid. These activists love to ban words in everyday common use—the term "cloud" for third-party computing sites is another example—and their reasoning always has some legitimacy. They accurately argued that the various regimes falling under the term "intellectual property" were too different to be lumped together. But we were not about to saddle our book with the label *Copyright, Patents, Trademarks, Trade Secrets, and Free Software*, so we had to generalize somehow, and Intellectual Property was the perfect term. There are professors of intellectual property at universities, law firms advertising their services in intellectual property, and other firmly established uses of the term.

I doubt whether our potential readers were deterred by our using either of the terms the FSF dislikes. The book sold poorly because, I suspect, programmers and system administrators love to talk about the topics flippantly but won't be coerced into actually learning about them. I idealistically thought that since numerous computer professionals complain all the time about the abuse of the patent regime or copyright system, they would want to understand the real principles at stake.

It's easy to dismiss concerns about these legal areas as fussy. Tim O'Reilly has regularly said that licenses aren't particularly important in free and open source software, and that building a healthy, responsive community is much more important to success. Although I could agree with his ranking—community is more important—the license is not a trivial matter. A failure to respect a license, trademark, or other aspect of intellectual property can lead to abuses that hurt free software and open source communities.

Nor is anything simple with these topics. The history of fair use is a case in point. Once, when talking to the O'Reilly company's own lawyer about some matter, I suggested that someone's use of a trademark might fall within the scope of fair use. "The

concept of fair use doesn't apply to trademarks," he snapped at me. "Fair use is only for copyright." This lawyer vanished from the company a few months later. I'm sure that any lawyer working for a publisher needs to understand trademarks, because we deal in them a lot.

O'Reilly was always the victim of flagrant copying, although we refused to take measures we felt were unethical to combat copyright infringement. We wouldn't wrap our books with Digital Rights Management, or what opponents of the practice like to call Digital Restrictions Management. When we forged a partnership with Microsoft Press to distribute their books on Safari, Laura Baldwin declared that our refusal to use DRM could be a deal-breaker, and after some grumbling, Microsoft Press management agreed to forgo it.

»»

DRM consists of techniques involving encryption and digital signatures to prevent people from using content without paying for it and obeying whatever rules the content producer has set up. If you bought a video in Europe and find you can't play it in the United States, you have run up against DRM. As the example shows, DRM encourages a lot of arbitrary restrictions, which in turn contribute to odd business models. Worse yet, DRM is often poorly designed and fails, so copyright holders have forced numerous laws to prevent tampering with DRM. The far-reaching impacts of these laws are even worse than the DRM itself. For instance, it has a chilling effect on a lot of research. Researchers have actually been arrested for offending the copyright holders.

The easiest way to mitigate against infringement is to provide content as a service and update it frequently, so that no single copy has much value. Thus, a side benefit of O'Reilly's turn to an online-first strategy in the 2010 decade was to create a bulwark against copying. In contrast to our earlier principle, I don't believe that freedom from DRM was considered during the move to our online strategy.

Slump (2000s)

Several impacts of computing and networking on society in the early 2000s became evident to those who followed technology. We saw entire fields transformed or eliminated, and jobs with them. I could tell that automation would eliminate more and more jobs in manufacturing, construction, and other fields that would seem protected from the Internet. I started talking to

friends and colleagues about the unhappiness this could cause, expressing concern over the role I might be playing by making the technologies easier to use.

All the people I talked to were reassuring. They may or may not have agreed that a worrisome social transformation was taking place, but they said that I shouldn't take any burden for it on myself. Even my brother Alan, who is both politically astute and highly spiritual, took this position. But now that the damage to society and the public sphere is so obvious, I wonder whether the question should be revisited. Was I making life better or worse?

It turned out that I myself fell victim to these trends. My magic touch for finding topics and signing authors abandoned me as Internet usage unfolded throughout programming. And although the 2000s and early 2010 decade presented many gratifying opportunities of which I can boast, I constantly felt pursued by the gremlins of doubt, depression, and distrusting my basic competency.

The contrast with the 1990s couldn't be greater. That was the period of the dot-com boom, when the door was wide open to fine texts on computing. I seemed to ride a surge of triumphant publishing campaigns. The two series I championed on Linux and MySQL flew off bookstores shelves. There was no difficulty generating proposals and signing authors on topics that reflected obvious user needs.

Shortly after 2000 this all fell apart. I still can't explain quite why. I was doing all the same things I did during the 1990s—maybe that was part of the problem. The publishing field was different, and I think everyone in publishing, journalism, or media felt it. No series of books would save us. To avoid the fate of so many publishers who were throttled by their own business models and vanished, we had to go beyond the books that had formed my career.

I'll offer two examples of failed projects to show the challenges facing me as editor at that time. Both examples involve an extremely popular database engine, MySQL, which delivered huge sales for O'Reilly in the 1990s and early 2000s. As editor of the series, I was always looking for new ways to milk the topic. I had formed relationships with numerous employees of the MySQL AB company (headquartered in Sweden), from the top of the company on down, and kept in close contact after Sun Microsystems and then the Oracle Corporation bought their company.

Sveta Smirnova came highly recommended by my lookouts at Oracle. She was responsible for troubleshooting there, as a Principal Technical Support Engineer, and wanted to write about how to troubleshoot problems in MySQL.

In any technical area—whether diagnosing the rumble made by your car or assigning blame for the Challenger space shuttle disaster—troubleshooting is the mark of the highest expertise. Most computer books offer you a set of steps, assuring you that if you follow these steps, things will come out right. Well, what if they don't come out right? What mischievous element hiding in your environment is foiling your best efforts? Troubleshooting is where programming meets Agatha Christie.

When it came to MySQL troubleshooting, Smirnova knew the answers. She could do exactly what every programmer or database administrator wanted: to start with the error staring them in the face, and take them on a quest that unmasked the criminal. I was in awe of her understanding; I'm sure she had saved millions of dollars for MySQL AB clients.

And I'm sure that, had she written her book during the peak of our MySQL sales, it would have become a must-purchase. As released in 2012, however, the book left hardly an impact. People were still using MySQL. But they were turning their attention to other technologies. We just couldn't spark excitement with the book.

The next example concerns a topic of pressing importance that got smacked down at the start. I had met two impressive engineers from MySQL AB who had written the book *MySQL High Availability* for us. These engineers, Mats Kindahl and Lars Thalmann, told me that MySQL AB, like O'Reilly, was highly scattered geographically. (I guess that Swedes don't have the luxury of building a world-class programming team using just Swedes.) So MySQL AB had to manage teams of people around the globe, and they saw other companies facing the same challenge.

»»

At that time around 2010, the rage among programmers was a spectrum of collaboration strategies that went by various names: agile programming, SCRUM, and eXtreme programming. These strategies all featured tightly integrated teams in a single geographic location. By talking face-to-face constantly with each other and with users of the software (or marketing people who supposedly could represent the users), these programmers could move fast. They built a tacit understanding and corrected each other's missteps informally.

Kindahl and Thalmann, by contrast, had learned collaboration techniques discovered by free software developers who lacked the luxury of geographic colocation. (MySQL itself is open source software.) When you're integrating code changes from Sweden, Canada, and Japan, you need the exact opposite of the Agile/SCRUM/eXtreme culture. Decisions must be explicit, and must be recorded. Each programmer must be granted autonomy and must be trusted to make key design decisions, although the results are certainly reviewed.

These policies about the location of staff concern all of us. They aren't just an abstract argument over how to do software engineering. The conviction that smart people need to be crowded together in order to learn from each other snowballs into the "innovation hubs" I criticized at the beginning of this memoir. Dynamic companies elbow each other while taking over desirable downtowns and flying in more and more recruits to glut the neighborhoods. The result is an overcrowding in places such as the Silicon Valley that's just as socially unhealthy as the corresponding starvation of the homesteads who lose their creative talent to these hubs. We're living every day with these problems in the Boston area.

I also argue—and am sure I'll draw heated rejoinders for doing so—that intense, colocated engineering practices hinder diversity in software engineering. Managers looking for new hires who can communicate easily with fellow team members, and trying to promote those who fit in most seamlessly, will consciously or unconsciously favor people who fit the team's current culture. It takes extra effort to incorporate diverse voices—can the high-pressure Agile/SCRUM/eXtreme teams afford to do so? This all depends on how much the managers care about diversity and their awareness about what it calls for. But I can't say that distributed teams solve the problem. These projects—including free software communities—need an explicit commitment to improve diversity too.

Knowing the achievements of distributed free software teams, and seeing similar distributed efforts all around me, I leapt on Kindahl and Thalmann's idea of a book about how to coordinate distributed programming efforts. O'Reilly management did not share our enthusiasm. They rejected our proposal, and I have to say that they were probably right. In the early 2010 decade, the computer industry was not ready for our message. A book like that would certainly be popular after COVID-19 blew apart all those intimate software teams. But when I was ready to cover the topic, few people recognized its value.

The idea was certainly percolating, though, because a similar topic captured O'Reilly's interest a few years later. Large companies with distributed teams were drawing on free and open source development techniques to get those teams to collaborate internally. When Danese Cooper, a strong open source advocate I had known for many years, approached us with the concept of InnerSource, the company assigned me to write a report about it. This report, "Getting Started with InnerSource", came out in 2015 and became the most popular report we had ever released.

So something had changed between my salad days of the 1990s and the parched 2000s. Why couldn't I continue acquiring as before? A lot of changes probably combined to quash my earlier success in signing authors.

The main change that wrecked my career, I think, may surprise you. It was the very movement I had championed and documented for years: free software. Programmers who came to understand free software and to witness its rapid success realized that they, too, could have a large impact on society by contributing to the software. In fact, free software could appeal to any sense of grandiosity a programmer might have. For a talk I gave in Tokyo in 2001, I discussed the fantasy of control that grips hackers as they code, documented by Joseph Weizenbaum as far back as 1976 in his book, *Computer Power and Human Reason*. What would be more gratifying to someone looking to have influence than to contribute to a project used by a hundred million people?

But influence is just a sweetener—perhaps an important draw, but probably not one that a programmer would be conscious of (unless they read the speech I gave in Tokyo). The more important attraction for programmers was the chance to be recognized. If you wanted to be a leader and influencer in programming, the way to do that used to be to write a book; now it meant writing software. Your fellow programmers would find the code you write more meaningful than a text describing what to do.

The publishing model in computer books was always based on converting programmers temporarily to authors to benefit their reputation. For years this worked magnificently. I heard from two authors who believed that the publication of their books allowed them to start their own consulting firms.

But if you could do just as well, perhaps even better, by remaining a programmer and contributing to free software, why go outside your comfort zone to write a book? And incidentally, royalties from most books were far less than one could earn at a conventional programming job, so financial compensation wasn't much of an enticement.

The Internet also changed advertising, undermining old ad channels such as journals. Self-promotion became the royal road to influence, and this permanently altered the power relationships of authors and publishers. Let's look at how the shift played out.

O'Reilly changed certain practices in response to the diminished reach of advertising and public relations. Although we usually found our authors through their active participation in technical forums, we used to ask them in our early years to withdraw from public participation while they were concentrating on their books. But as Internet communities became more and more important for gaining a following, we reversed our position and encouraged authors to stay connected to these communities.

Although at first that change was simply opportunistic, this gargantuan contest for visibility eventually hardened into a strategy—one used by all publishers. They started to ask potential authors about the sizes and composition of their "platform", meaning how often they posted to blogs or forums, how many people followed them on social media, etc. Publishers were

backhandedly admitting that their own marketing efforts fell short in selling a book; the author was explicitly asked to do their own independent marketing.

No surprise, then, that lots of authors decided to self-publish. They could well argue that their platforms were promoting their publishers more than the publishers were promoting the authors. O'Reilly's move to an online subscription model was our response to this threat: Our data-enhanced service could link up readers to appropriate content much more reliably than conventional advertising had done.

I could cite lots of other reasons that all my old wiles crumbled when trying to recruit authors. The vast firehose of Internet content enticed people with the sense that they could get any information they needed for free (although most came to realize that quality was highly questionable), so sales of books were dropping. Technology started to change so fast that authors could barely keep up even as they were writing, and the pressure to finish fast was increasing. I'm speculating about all this, of course, but I think a lot of factors came into focus like sun's rays through a convex lens, and the result was a career in flames.

It took me a couple years to assemble this explanation for my change in fortune. At first, all I knew was that authors weren't returning my email messages. Or they might submit an outline but never follow up and sign a contract. Some even signed contracts and never turned in a single word of content—probably they started writing, found it more work than they thought worthwhile, and dropped silently away.

The slump from about 2005 to 2015, which more or less corresponded to our company's Fawcett Street location in Cambridge, constituted probably the most difficult time of my life up to that point. Things that used to work for me no longer did. I was like so many other workers alarmed at their status during times of great economic change: I was doing my job competently the way I had always done it, and finding myself a failure nevertheless. I can empathize with people around the world who protest modernity after finding themselves driven from their customary positions. The closings among publishers in this period went along with the decline of journalism and other fields recoiling from the impact of the Internet.

My commitment to free software from the beginning of my O'Reilly work was so strong that years after we retreated from treating free software as a special focus of our week, I was generally identified by other staff as the "open source editor" as they directed authors my way. I tried to build peer-to-peer into a similar homestead on which I could settle and peddle my work, but the movement was not successful enough. I started to realize this around 2005. At the same time, I could see my status as a source of

information and authority on computing trends slipping away. Invitations to conferences continued to come in for a while, but I eventually saw a decline there as well.

To be sure, many wonderful moments lay ahead. My work with the Peer to Patent project, which I had heard about first at a Chapel Hill conference about copyright and free culture, earned my articles publication in The Economist and Communications of the ACM. My investigations of health IT and open government opened many doors and introduced me to wonderful people who became new communities to me. In 2007 I edited one of my most successful books, both commercially and reputationally: *Beautiful Code*. It led to an entire series.

A survey of my blog postings over those years shows intensive activity. I continued to cover free software while adding copious amounts of material about government and health IT. I was traveling extensively, conducting video interviews, and signing up people for all kinds of schemes that didn't pay off.

This period was filled with false starts, by both me and the company. Government work and health IT failed to produce revenue. I marshalled O'Reilly around an audacious comic book project, Hackerteen, and we just lost money.

Thanks to shifts in both technology and the publishing industry, every successful book that I tried to update failed in the new editions we put out during the 2001-2010 period. This includes our Linux kernel books, which could not keep up with the kernel's speed of development and increase in complexity. Embedded systems, which had produced several lucrative books nestled in a pleasant little series, couldn't sell any longer. My highly regarded MySQL series fell apart and new offerings in the series failed to sell. (At the same time, Oracle bought the MySQL company and decided not to keep sponsoring our conference, which I had covered every year.)

So the decade from 2005 to 2015 was therefore a time of confusion for all. In addition, for me it was a time of shame and vulnerability. I certainly wasn't fulfilling the role of a senior editor, who is responsible for looking ahead, for setting the strategic direction of their area of activity—and not incidentally, for creating profitable books.

I don't remember when I was promoted to senior editor. When I came to O'Reilly, there were very few editors and all of us took on a grand buffet of responsibilities: We interacted with the communities we were documenting, explored new possibilities, used the technologies we were writing about, and forcefully stated our opinions. The appellation "senior" barely registered with me. I noticed that many editors—at O'Reilly and at other publishers—put the title "Senior Editor" on their business cards, but I just

stuck with the simple title “Editor” because I figured that the difference between an editor and a senior editor was an internal company matter, not relevant to outsiders.

At the beginning of this decline, I had little idea of the analysis I developed later. I just felt the ground falling away. With teens at home, my wife was busy with other things and didn’t really understand the position I was in. No one outside of work could appreciate what was happening, so I sometimes confided (and even complained) to coworkers. I admit it: I was seeking empathy and emotional support in a business environment, and even in the loose and accommodating O’Reilly culture, I was probably acting inappropriately. Perhaps I am still doing it here by writing up my memories. But it’s necessary to write about this period to convey the devastation felt by people dislodged from their comfortable, thriving context. In my case, the feeling went on for as much as a decade.

O’Reilly management was changing its strategy in reaction to market shifts too, and started rejecting more of my proposals (when I was lucky enough to get a proposal from an author) than they accepted. The company was squeezed from all sides, and so was I. Although the company could supplement its books and online articles with other forms of content such as video and online training, I had nothing special to offer in those media. The equivalent of book editing in video would be rather eerie: It would have meant watching an expert deliver their material in front of us and keeping up a running commentary: “Substitute this word for that…turn this way and raise your hand…look straight at the camera…now summarize the topics for the past ten minutes…” We are not film directors, however. And I was never asked to translate my editorial skills to the video realm even though I’ve delivered a number of my own successful presentations, some caught on film.

I might well have joined the ranks of the obsolescent and left O’Reilly with a severance package if Mike Hendrickson had not tapped me for his sponsored content program. In many ways, our shift from books to other forms of content was a great relief to me. The company’s scrutiny of schedules for our books softened. No longer would I be asked to provide detailed weekly updates to an editorial or production manager. Although I would note that a book is slipping and take decisive action to mitigate the problem, I would never again have to spend agonizing hours on the phone about it.

Influence? (2000s)

People often professed to me that they saw an indelible association between me and O'Reilly. This was quite flattering, but less and less accurate as years went on.

These people might be basing their impression on bombardments of blog postings and policy suggestions I made as O'Reilly got heavily involved in government and health IT. Or outsiders might be thinking back to my round of keynotes and conference presentations following our book and conference on peer-to-peer. Many people with schemes of collaborating with O'Reilly tried to gain entry through me (and sometimes I could successfully enroll them in our work).

Many remember me as a champion of free software, which I expressed through more than 400 articles, along with my books on the Linux kernel and other important free software projects. I was certainly at the center of corporate planning around open source, and had influence as late as 2007 through my friendship with author Steve Souders and our proposal for a conference on web performance, which I will describe later.

But the period when I really felt empowered was the 1990s, as part of that small team of editors who met once or twice a year during the early years of O'Reilly's publishing business. This was a unique group of unconventional people with unconventional thinking. I don't believe a single editor during that period came from another publisher. We were programmers, technical writers, or geeks with various backgrounds attracted to the joys of explaining difficult concepts. We were constantly talking to one another—or so it seems from this far remove. We often grumbled that we didn't coordinate our efforts enough. But our camaraderie and shared vision from those young days are things that the company tried to recapture many times since, with only moderate success.

This editorial vision dissipated in three ways, and my sense of being an influencer diminished along the way. First, the locus of decision-making shifted away from editorial meetings until they eventually ended altogether. The last editorial meeting for which I have notes was 2011. The company set up other institutions, such as a steering committee, that drew on experts from many branches of the company. And clearly they did a good job, because O'Reilly continued to prosper and draw praise from everywhere. I simply never participated in those institutions.

Secondly, the company started to hire professional editors from other publishers instead of hiring programmers or system administrators. Starting around 2000, O'Reilly started to look a lot like other publishers. But we still prioritized inspiration and wanted to play a role in changing things, as evidenced by our Government 2.0 and health IT work.

Finally, as the options available in publishing narrowed, O'Reilly's strategy took on a focus that ruled out quirky editorial projects such my 2007 best-seller, *Beautiful Code*, or my 2009 contribution to hip computer culture, *Hackerteen*. It had become clear to management that our survival called for a subscription model, thus limiting our product line to material for professionals who would come back for new learning. The company then coalesced further around a business-to-business platform, where content was directed to showing results. By that time, although I had the occasional chance to appear on the O'Reilly web site, I was in no sense a spokesperson or important entry point for the company.

Web performance and Velocity (mid 2000s)

The key to finding successful publishing topics is gaining a place of trust among the most successful people in a field. These people manage to balance the implementation of the best current technologies with a passion for advancing the field. I occupied a position of trust and companionship with such people for a couple decades, and enjoyed the momentum that made possible. That is, each successful project would lead to the next.

A good example is how I discovered and promoted the field of web optimization.

The story starts with a book on performance in a somewhat different area: relational databases. O'Reilly had a small set of books in the mid-2000s about the most popular free software database, MySQL. But the series really took off in 2004 when a database administrator named Jeremy Zawodny approached me with an idea for a book about how to significantly speed up queries. Aided by co-author Derek Balling, Zawodny put together *High Performance MySQL* and released it to a hero's welcome by the community. Another set of star authors updated it, and the second edition continued to sell well. It then declined in interest, along with the rest of our MySQL offerings.

But in the meantime, I had made a leap to an even more lucrative area through a personal connection engineered by Zawodny. Working at Yahoo!, which was still an important force in the computer industry, he took notice of the interesting things that a colleague there was doing to speed up the loading of web pages. Zawodny brought me to Yahoo! one day, where Steve Souders dazzled us with a demonstration of tools that delineate exactly how long different elements of web pages take to display.

Why was web performance such a pressing issue? In those days, network bandwidth was much lower. It was common to wait several seconds for a page to load. Web designers thirsted for anything that could shave half a second or so off the "World Wide Wait",

as cynics called the Web.

Souders found ways to cut some loads down to a tenth of their original time. Some of the fixes were trivial, such as moving elements from the top of a page to the bottom. Other tricks were more complex. And there, as we marveled at the graphs in Souders's office in Sunnyvale, California, the book *High Performance Web Sites* was born. After its stunning roll-out, Souders proposed a conference called Velocity, and pulled together experts in his field from several large companies to guide us.

After connecting Souders with our management and seeing his initiative come to life, I approached the first Velocity conference with excitement. I was booked on a flight from the tiny Manchester, New Hampshire airport instead of Boston's Logan, probably because my family and I had planned a New Hampshire vacation around then.

As soon as we boarded the small plane, we became spectators to a freak hailstorm that passed over the airport. It lasted about fifteen minutes, but was enough to totally disrupt the airport's schedule. They announced that our flight could no longer take off, because other planes had priority, and that there would be no other flight on that route today. I managed to find a shuttle bus to take me back to Boston.

Because Velocity lasted only two days, and because I would miss about two-thirds of it if I went a day late, I just canceled. Other people took over leadership in my absence. Even though I continued to work with Souders and do another book with him, I never played a leading role in our Velocity efforts. It seems like almost a sign from heaven: One fifteen-minute weather event that disrupted a career. This incident illustrates once again the central importance of conferences in O'Reilly's approach to communities and publishing—the whole web performance effort revolved around the conference, and therefore around the staff who could attend the conference.

Technological improvements on the Web, including faster networks and the growing dominance of content management services such as Akamai, have reduced the importance of planning a web site around performance. But similar issues have come up subsequently around mobile devices, which also combine the dual challenges of low bandwidth and impatient users. I tried repeatedly to find someone to write a book like *High Performance Web Sites* for mobile apps. But by this time the relationship between publishing and technology had shifted, in ways I described earlier, to devalue books. Talented programmers were less interested in writing, and showed less tolerance for the stresses of being an author.

As for conferences, I was lucky enough to have a second chance. Peer-to-peer would give me an even bigger space than Velocity to move the computer field ahead.

Peer-to-peer (early 2000s) and the emergence of user-generated content

The most exciting phase of my tenure at O'Reilly, and of my entire working life, was taking a role as a representative and exponent of the peer-to-peer community. This section is the intricate story of that important movement and how I brought it within O'Reilly's sphere of influence.

By 2000, I was enjoying the height of participation in the free and open source movement, which offered immense opportunities to both me and O'Reilly to make money, proclaim opinions, and embed ourselves in fast-moving community efforts. I was in the open mindset when I started hearing some buzz about a strange new file transfer service, and decided to look more deeply into it.

Some background is called for here—an unusual potpourri of topics about the Internet, the music industry, copyright law, and my own obsessions.

⋙

Nearly everyone with an Internet connection (or so it seemed) showed interest in getting music over the Internet, and the near zero cost of reproducing and transferring music over the Internet made sharing an irresistible temptation. No streaming services existed at that time—just individual files containing songs in the ubiquitous MP3 audio format.

Little would hold people back from illegally sharing copies, except guilt over depriving artists of revenue. Even guilt-prodded listeners might know that recording companies absorbed the vast bulk of the revenue, a long-standing scandal in the popular music field. Who had sympathy for recording companies? And who felt a strong commitment to copyright?

The minor barrier that held back rampant sharing was a lack of knowledge about who had the MP3 file you wanted and how to reach them. These logistical details kept enough people away from file-sharing to mollify the copyright holders. They could still sell enough CDs to turn a profit.

In short, there was a delicate balance between technological capabilities and the old methods of doing business. Whenever this is true, advances in technology (as well as the resourcefulness of smart people starting new businesses) disrupt the balance. In the late 1990s, the disruption consisted of a service called Napster, which connected people who didn't know each other so that anyone in the world could send MP3 files to anyone else.

I did not try Napster myself, because I worked for a publisher and maintained some respect for copyright—but to be honest, I also did not want any of the pop music people were exchanging. By all accounts, the interface was horrid. Still, using Napster was simple enough: Sign up, upload the list of MP3 files you had converted (ripped, in common parlance) from your CDs, and search for a keyword that would turn up files other people had offered. Note that Napster didn't keep any files on its site—only lists of files offered by other people. This kept it lean. Its own bandwidth needs were moderate, and the people who were sending and receiving files did so directly, keeping their own bandwidth use within tolerable limits. Napster's delegation of file transfers to individuals at the edges of the network heralded more radical architecture changes to come—we'll see how that led to peer-to-peer.

Soon, Napster was connecting an estimated 80 million users. Now the music industry had to take notice.

Law professor Lawrence Lessig wrote a book in 1999 called simply *Code*, in which he discussed how software could hand unprecedented power to companies and governments. His observations are no mere philosophy; they become central to the culture battles I'm discussing here. Lessig's book productively distinguished four approaches to controlling people, such as passing laws or promoting norms. (Norms don't seem to have the hold on people they used to.) The approach Lessig focused on was code, which could actually prevent people from even committing an infraction. Code is to law as a speed bump is to a speed limit. A sign posting a speed limit merely enjoins you to obey a law, whereas a speed bump damages your car if you disobey. Code controls us too, more than we ever think.

Lessig became a leader and even icon of computer activists like me (I reviewed drafts of the second edition of his book) and he participated in some critical historical movements among computer activists, such as founding Creative Commons. That organization took off from Richard Stallman's brilliant free licensing idea for software—which Stallman called copyleft—and extended the concept to media. Lessig was an integral part of the free software and free culture movement before he decided to devote himself (tilting at windmills, in my opinion) to the distant goal of removing the corrupting influence of money from elections, and eventually even running briefly for U.S. president.

The music industry used several of Lessig's approaches in a full-blown war on MP3 sharing. Many network administrators on college campuses used technical means to identify and then throttle Napster downloads, not so much because they cared about the revenue of the music studios but because the network traffic was burdening their routers. The music studios tried to work with telecom and Internet providers to do something similar, but the intrusive procedure was too creepy in the public's eye to carry out. The music studios and some of the musicians also invoked norms, asking their fans to give up the practice of downloading music to which they were becoming addicted, in honor of the musicians who needed to make a living.

Ultimately, the recourse had to be a legal one. Clearly, copyright was being violated to the point of torture through Napster. But who was guilty? Napster wasn't storing any music, wasn't transferring any music, and wasn't making money off of selling music. (I believe it made money from ads, the nearly universal business model of web sites since the mid-1990s.)

Individual Internet users were clearly guilty, and occasionally the copyright holders would come down hard on them. I don't remember whether this practice started in the time of Napster or during the next phase of music sharing. The vendors would monitor publicly exposed information to identify particularly prolific music sharers, and would then haul them into court for serious monetary damages in order to make warnings of them. This was termed a whack-a-mole strategy, and was not a sufficient deterrent. Many of us in the Internet community tried to suggest to music studios that dragging their musicians' biggest fans into court wasn't great public relations. We also questioned whether the decline in music sales was caused by online file sharing or trends in the music industry itself.

Finally, copyright holders nailed Napster on the obscure doctrine of "vicarious and contributory infringement". It had been part of copyright law for some time, and had many real-world applications. For instance, suppose you set up a flea market on your property and bring in a bunch of vendors who offer bootleg CDs or videos. The police will pick up the bootleg vendors for sure, but they will also prosecute you because you knowingly let them sell their wares. Incontrovertibly, Napster was doing the same thing over the Internet. And thus, in July 2000, a judge shut Napster down.

At that time, I was producing a monthly column for an online journal called Web Review, run by the adventurous and highly effective web expert Molly Holzschlag. While her site devoted itself mostly to practical guidance for web designers and

programmers, she enjoyed publishing my column on policy and social issues related to the Web. I called my column Platform Independent, a play on the words used by the inventors of the Java language. They boasted that Java could run on all platforms without compilation (and therefore was platform independent), whereas I commandeered the term to show that I would maintain my own opinions without regard to other people's political platforms. In those voluptuous dot-com days, Holzschlag could afford to pay me $300 an article.

On the day the court ruling against Napster came down, Holzschlag wrote me to say that if I could produce a thoughtful comment on the case for Web Review within 24 hours, she would pay me a thousand dollars. I wrote a nice piece titled "The Napster Case: Shed the Baggage and Move On", where I highlighted subtleties ignored by the mainstream media (who jumped on the issue like sharks), and expressed my hopes that the precedent set by the case would not hold back Internet innovation.

Indeed, the crushing of Napster was not the end of the drama. It perhaps ended the first of three acts. Peer-to-peer came on the scene at the next curtain rise.

»»

In 2000, news reports started coming out about new file-sharing networks called Freenet and Gnutella. With different protocols, they both solved the legal problem that slew Napster: They had no central point that could be sued. Instead, they used complex techniques for distributing content from computer to computer and for letting people ultimately discover who had the content they wanted. For practical reasons, each user of the system saw other computers up to a limited horizon—the services did not try to reach the entire Internet for all users.

The creators of these clever protocols explicitly presented them as workarounds that exploited holes in the copyright doctrine to which Napster had succumbed. The technologies also had legitimate uses, because they were a good way to exchange material generated and owned by the sender. (A later implementation of the concept, BitTorrent, became a major legal channel for distributing free software.) So the people who coded and distributed the software were protected from lawsuits by the Betamax decision, a long-standing court case saying that the makers of a technology are safe from copyright suits so long as it has "substantial noninfringing uses".

The release of Freenet and Gnutella had the flare of publicity, if not an open provocation to the powers that be. So it came to the attention of the popular press, which had mostly ignored the Internet till recently (and were soon to find how destructive it could be to their own business models). Once already, the press had been dragged into trashing the

> Internet. This trigger was the fight over pornography embedded in the Communications Decency Act, which sounds like something from the nineteenth century but was actually passed by Congress in 1996 and soon struck down as a violation of the First Amendment. The Napster case raised the media's leeriness to a state of panic. And now suddenly—Freenet and Gnutella.
>
> So the news media went berserk. They had breathed a sigh of relief when the courts shut down Napster, but they realized that technology left no haven for old ways of thinking. They presented Freenet and Gnutella as an existential threat to traditional media.

While everybody else was debating the rights and wrongs of file transfer and whether the new technologies were legal, I took off in a different direction. I was sure that any technology that could scare so many people must be technically powerful. I read whatever technical material was available about Freenet and Gnutella. I extracted enough from this material to write a blog posting for the O'Reilly site in late 2000 with the title "Freenet and Gnutella Represent True Technological Innovation". The title was almost click bait, not only challenging the conventional narrative but flaunting my disregard for that narrative.

Even though I expected some notoriety, I was surprised a week or so later when Allen Noren, who was running our blogs, told me that this posting had pulled in more views than anything else ever published on the O'Reilly web site. We agreed that more coverage of the Freenet and Gnutella phenomenon was called for. Others in the company noticed as well. Tim O'Reilly encouraged me to put together a book about new trends in decentralized networking, and corralled potential sources from his own Rolodex of august contacts.

Within a couple months, I assembled a top-drawer collection of technologists, along with a few social scientists to add that extra dimension, for a book on decentralized technologies. Freenet and Gnutella represented just one branch of an experiment in decentralization meant to shatter the current, restrictive structure of the Internet where powerful servers determined what their clients could do. This network architecture, client/server, had taken over during the past five to ten years. Peer-to-peer was a direct challenge to it.

Decentralization went hand in hand with the openness represented by free software. However, a few companies were designing proprietary decentralized technologies too, claiming to apply this old-new model to numerous business problems, just as many years later companies adopted blockchain in an attempt to solve everything in a new way.

I knew our book challenging the conventional hierarchy on networks would jump into a hot debate by people up and down the ranks of corporations and government. I also knew our book would be the only one worth reading. Having taken up the project promptly and swarmed the computer industry with solicitations before others thought of the topic, I knew that everybody who had anything worth saying was either writing for us, or was too busy to write at all.

Although we tossed around many titles for our anthology, by the time we rushed the draft to production someone had popularized the term "peer-to-peer". It was obvious that this must be our title. And I found the term gratifying, for the Internet was founded on systems communicating with each other on an equal basis. Establishing connections was called "peering" from the beginning of the Internet. By the mid-1990s, only a few very powerful computers in upper tiers were still peering. The rest established contractual relationships with bigger networks in higher tiers and smaller networks in lower tiers. Still, communicating as equals is a fundamental Internet concept. The peer-to-peer movement around 2000 applied this concept to voluntary networks created on an ad hoc, volatile basis on top of the conventional Internet protocol stack.

»»

> The new peer-to-peer movement challenged a lot more than models for distributing content. At its most ideal, it represented a new way of tying together people without centralized mediators. It was also a new approach to the bandwidth problems hotly discussed around that time: Peer-to-peer exploited increasing Internet bandwidth, while offering intriguing new ways to use it. (Essentially, researchers found, peer-to-peer networks reduced the load on individual servers while increasing overall traffic on the network.) Lots of considerations lay behind a blog posting I wrote in September 2000 titled, "Peer-to-peer starts to make the Internet interesting again." I was suggesting that we were all tired of what client/server had to offer.

At bullet speed, without sacrificing quality, I got the 450-page *Peer-to-Peer* to print in time for a conference of the same name that O'Reilly hastily called in February 2001. The 19 chapters of the book fell nicely into three sections: context (the social and philosophical underpinnings), projects (Freenet and Gnutella along with others), and technical topics. I expected that reporters, government employees, and business managers would read the first one or two sections, whereas those with the stamina for the technical meat would go on to the third.

Because it was a multi-author anthology, my name appeared on the cover. This was standard practice for publishing: I had assembled and edited the whole collection, although the only piece with my by-line was the preface. It was a fortuitous move, though. Having my name on the cover of the book led to great opportunities.

We held the Peer-to-Peer conference at a hotel in downtown San Francisco, eschewing the traditional convention sites, and limited registration to about 400 people. At the standard convention center, attendees cycle between cavernous, characterless meeting rooms and even larger, characterless corridors, bleaching inspiration from the attendees' minds and souls. In contrast, navigating through the elegant twists of the San Francisco hotel, one was constantly wondering whom or what one would come across next.

Backpack handed out as swag at the Peer-to-Peer conference

The conference sold out right away, as this was still the time of the dot-com boom. We brought early copies of the book to the conference for sale to the attendees. (Remember that point for an intriguing tale to come.) The official release was in March.

The first organization to pick my name off the cover of Peer-to-Peer and call me up for an interview was the radio show called The Connection, on Boston's WBUR station. Under the dynamic host Chris Lydon, The Connection had achieved renown. Lydon was long gone (because, I suspect, he demanded a salary of a size commensurate with his fame and influence), but being called on the show was still a big honor. I did a good job, and ended by advising artists not to worry what the Internet would do to their careers. In this I was probably a little Pollyannish, but I stand by my basic message on the show: The Internet presents as many opportunities as it takes away, and it is up to us to find the places where we can live off of its value.

Lydon, by the way, went on to found an Internet show dedicated to ground-breaking trends. He called it Open Source, which I found gratifying even though the choice demonstrated a lack of understanding about what open source really is.

One amusing copyright incident accompanies my appearance on The Connection (copyright is never far from everything nowadays). For our book cover, our designers took the famous picture of Adam from Michelangelo's Sistine Chapel, omitting the face and hand of God, and flipped Adam horizontally so that the cover showed two Adams reaching out to each other. I thought the concept demeaned Michelangelo's masterpiece and trivialized the book's serious concept, but went along with it. The staff at WBUR liked the two-Adam picture and put it on their web site to advertise the segment where I spoke. Within hours, our designer Edie Freedman phoned WBUR and warned them to take it down. She explained that because a cover designer had altered the Michelangelo, the picture was not in the public domain but belonged to the designer and could not be displayed without the designer's permission. However, she said, they could display the entire cover containing the picture, which in relation to my appearance would constitute fair use.

My radio appearance was only the start of the fame that rolled out from my listing on the cover of Peer-to-Peer. I was invited to lecture in settings ranging from Japan to the illustrious FOSDEM conference in Brussels. The peer-to-peer idea as an independent movement then faded, but it had a long-term, unanticipated influence on the next phase of Internet development that changed the world.

I keynoted a system administration conference in Tucson, Arizona, where I urged the attendees not to block peer-to-peer traffic. Then I got a great opportunity to fly to Brussels and deliver a talk on peer-to-peer at FOSDEM, the leading free software conference in Europe. Staff from O'Reilly's office in Britain were there—people who had helped me innumerable times over many years, appreciated my work, encouraged me to carry out bold suggestions—so I could put faces to names and share thanks with them.

Richard Stallman keynoted at the conference, joking around with his well-known Saint IGNUcius halo. I was surprised by his presentation, a generic exposition and defense of free software. My impression was that all the attendees were avid members of the movement, being users of and perhaps contributors to free software. But Stallman told me, in his review of this memoir, that many FOSDEM attendees failed to understand the ethical and social significance of the "freedom" in free software. So he believed that conveying this in his keynote talk was of critical importance.

My own talk at FOSDEM was not well attended because it was placed opposite one of the most celebrated free software developers of that time (Miguel de Icaza, if I remember right), and in fact I was not particularly happy with my own delivery. I think now that I could have pursued my ideas further and added more practical ideas to the talk.

A note on de Icaza is of interest. He had amazed everyone by producing a free clone of the complex Microsoft C# environment. The goal of this project, called Mono ("monkey" in Spanish, but probably a name indicative of its unifying ideal) was to let people interact with the proprietary Microsoft .NET at every level, using nothing but free software. Microsoft at the time still harbored a visceral fear of and antagonism toward free software, but eventually they welcomed Mono and collaborated with the project. Unfortunately, de Icaza couldn't get a sustainable business going through free software—discovering what hundreds of other failed companies had found out—and ultimately announced that his next project would be proprietary.

The biggest boondoggle to come my way was a trip to Japan, a direct invitation from the head of the O'Reilly Japan office, whose name was Arihiro Iwata. When speaking English he went by the name Alex.

For this trip I prepared not just one but two new speeches about peer-to-peer. First I was to keynote at a large conference (whose exact purpose I never understood) called Info-Tech2001, sponsored by the Kansai Institute of Information Systems in Osaka. Then I would visit the O'Reilly Japan office in Tokyo and give a talk before a select group of political science professors at Senshu University. I don't know how Iwata wangled these invitations. I am also not sure that he asked for two separate speeches. Following my editorial principles, I may have decided that the two audiences had different needs and interests, and therefore that I should write two speeches. That may stand as my basic principle for both writing and editing: Know your readers and deliver what they need.

Somehow, in the three months leading up to my trip, I managed to write both speeches and study some Japanese. I learned the two alphabets, Hiragana and Katakana, including how to write my own name, along with a bit of grammar and some useful words. Oh, and also carried out my editorial duties at work and raised my kids. The responsibilities did seem to slow down my blogging, though: My records show only three postings during that whole time.

After I mentioned to my relatives that I would be going to Japan to give speeches, I got an odd request from my sister-in-law Coletta Youngers, who was working on human rights for the Washington Office on Latin America. She was incensed that former Peruvian president Alberto Fujimori had fled charges of human rights violations by taking up residence in Japan, which showed no interest in looking into the charges or requiring him to go back and face them.

So Youngers asked me to work into my speech about peer-to-peer technology an appeal to my audience to extradite Fujimori. This would seem an odd fit, but I came up with: "individuals can do bad things on the Internet and then disappear, leaving behind the problems they've caused and popping up somewhere else. How will peer-to-peer systems deal with their Alberto Fujimoris?" Youngers thought this hilarious and apt. Sensitive to the mores of cultures I knew little about, I asked my sponsors in advance to check this passage and let me know whether including it would offend anybody. They approved it, and I delivered it with gusto in Osaka. I don't take any credit, however, for the extradition order issued by Japan many years later.

Iwata treated me royally. He paid for first-class seats on flights to and from Japan, and took three days off both from his office and from his wife (whom I never had a chance to meet) to be with me every waking minute and guide me through the geographic and cultural labyrinths of his country.

I had learned a number of important points about Japan some twenty years earlier when working briefly for Hitachi. I bowed as gracefully as I could upon meeting people. When the professors in Tokyo presented me with their business cards, I accepted them with both hands and laid them out in a vertical row in front of me as I prepared to speak. I even referred to one of the professors by name to acknowledge an idea he had given me during our initial chat. (I hope I didn't insult all the other attendees by leaving them unnamed.)

But I was not prepared for everything. During the dinner I attended at the Kansai Institute on the day before my speech, one of the organizers unexpectedly asked me to say a few words. I simply introduced myself and said I was honored to be invited. After I stepped back, someone else came up and offered a few minutes of talk in Japanese which I recognized was a follow-on to my speech. Iwata told me later that I was expected to say a lot more, and that proper protocols were rescued by this other man who filled in what I was supposed to have said.

Our schedule in Osaka left some time for sightseeing, so Iwata treated me to one of the travel highlights of my lifetime: three of the huge gardens of Kyoto. I had visited the Japanese gardens in Portland, Oregon, and in San Francisco, but they were like children's models next to those of Kyoto.

We then took a bullet train to Tokyo. We couldn't get seats together, but I had the tremendous luck to sit next to a doctor who had lived in the United States and therefore could chat comfortably with me in English. He actually invited me to visit, a lovely gesture (whether or not he really meant it), but I could never take him up on it.

The Senshu University talk also went well, so far as I could tell. I alternated speaking with Iwata's son, who was a professional translator and translated my speech as I went along. I had to cut at least a third in order to fit within the time allotted.

Iwata took me to meet a technologist and successful entrepreneur who later became famous for good and bad reasons: Joichi Ito. We had dinner at a nice French restaurant, which I was looking forward to because I had had no one to talk to for days except Iwata, and looked forward to speaking French with the waiter. But it turned out that the waiter at this French restaurant spoke no French. With Ito, we carried on an animated discussion of the state of the Internet (although the details escape me).

Ito played many important roles later in computing, including stints on the board of Creative Commons and ICANN, before spending many years as director of MIT's highly visible Media Lab. MIT got involved in a scandal with a donor, disgraced billionaire Jeremy Epstein, and Ito resigned from the Media Lab.

In the O'Reilly Japan office, I was asked to sign Japanese translations of the book on Make that had been my very first work for O'Reilly, some 12 years earlier. They were impressed that I signed my name in Katakana as well as English. One staff person praised my handwriting, which I found dubious (because my English handwriting is an awkward scrawl), but I decided she must have been sincere because she could have chosen to praise me another way. Simply knowing my name in Katakana showed that I had taken considerable effort to honor their language.

Another staff person picked up a bunch of roses that had been sent for me and held it up to her face, asking me flirtatiously, "What looks better, me or the flowers?" Thinking fast, I answered, "You go well together."

Iwata rounded out my Japan experience with a mass at a Catholic church for a professor of his. I believe Iwata himself was Catholic. The mass was conducted in Japanese and took a long time, and I'm embarrassed to say that I fell asleep—Iwata had to prod me awake.

The only other regret I have about these three days in Japan is that, while Iwata took me to many restaurants, we never had sushi. I mentioned this to him, and he explained that he didn't like sushi. His overwhelming hospitality washed away any disappointment I had.

My keynote at the Kansai Institute was awarded an honorarium worth some $1,400. But in those days, converting the honorarium to a currency I could use was a convoluted affair. Iwata took me to a bank, which charged fees, and probably some

government fee was extracted as well. At one point I was handed a wad of Japanese cash representing a huge sum, and I carried it about me nervously, feeling like a drug dealer. By the time I deposited my earnings as dollars in my U.S. bank, the charges and conversions had reduced it to about $200. I didn't mind losing my payment because I had gotten so much other benefit from the trip, but I said, "Somebody's making a lot of money off of the global economy." I didn't know then how true this was.

My trip ended with another interesting moment. I knew that my child Sonny was playing viola in a concert by their youth orchestra on the day of my return, and I had assumed I would miss it. But arriving at Boston's Logan Airport in the early afternoon, I realized that I was just in time to attend the concert if I went straight to Boston University instead of going home first. I asked the taxi driver to change direction, and dragged my big luggage into the college a few minutes before the concert. People could hardly believe that I had just disembarked from a flight from Japan.

These stories pretty much sum up most of the ways I milked the opportunities around peer-to-peer. I believe I was a worthy emissary for the movement. Other people possessed a deeper knowledge of one technical topic or another, but I had the view from the balcony that came from coordinating so much of the communications around peer-to-peer. The role I got to play lasted from late 2000 through about 2005, when the problems of peer-to-peer communication loomed large enough to drive the concept back into obscurity.

»»

Even as we were working on our book in late 2000, fundamental design problems with peer-to-peer were becoming evident, and were even aired frankly in the book. I realized that pure peer-to-peer couldn't handle basic elements of addressing, coordination, and trust. I wrote an article for O'Reilly explaining those barriers in 2004. Even Gnutella (which long outlasted Freenet as a large-scale network), had to abandon the pure peer-to-peer architecture and promote some systems to superhubs in order to connect users at endpoints effectively.

The parallels between peer-to-peer and blockchain are interesting. Both excited the computer field because they promised to re-examine nearly everything that was taken for granted about coordinating people. The two trends were extremely broad, trying to cover everything people do. Unfortunately, they were developed with a deliberate disregard for the drawbacks introduced by their precedent-breaking designs.

An incident on Amazon.com which I found amusing—although I could just as easily been infuriated—underscored the weaknesses of trust inherent in peer-to-peer, and the Web as we know it today. The very first Amazon review of the Peer-to-Peer book

gave it the lowest possible rating—just one star. The reviewer openly admitted they hadn't read the book, but took a look at some online material and claimed it didn't go into much depth. In other words, Amazon.com's open reviewing system allowed a random, anonymous individual with no knowledge or competency to harm the reputation of my book. Some of the authors on the book asked whether we could get Amazon to take down the review (probably not), but I said, "Let's leave it up. It's a good example of the problems we've identified with peer-to-peer."

My writings that warned about the effect of malicious trolls on peer-to-peer systems apparently weren't read by the managers of the popular social media sites that were started years later. Perhaps they would have been prepared for the manipulations of elections and public opinion by governments and sleazy political organizations, which use the same tricks as that Amazon.com reviewer.

Peer-to-peer did offer me one last perk, quite an unusual one. A startup called GNUCo was trying to produce a commercial service to serve a huge number of different retailers. There was no way they could scale up to the extent they hoped to achieve if they had to keep up with every product change at every company. They figured that peer-to-peer technology could solve their problem. Wanting some confirmation, they offered me a trip to Atlanta to discuss their needs.

So I flew down and spent a couple days in meetings. Their platform was truly interesting and novel, and I made suggestions about what would be viable. I was impressed also by their impulse to give something back to the free software community from which they had adopted so much of their platform—the community that underlay the very name they chose for their company, honoring the GNU free software project.

I identified a neat dividing line between the bottom half of their platform, which might be useful to many people in many contexts, and the top half that was of interest mostly to them. What I suggested to them was a kind of open core strategy, in which they opened the bottom half as free software. I don't think they ever did this, and of course, they disappeared from the face of the Earth shortly after my meetings (I did get all my expenses paid, luckily), but it was an interesting experience that demonstrates the power of simple memes such as peer-to-peer.

Not only did peer-to-peer peter out as a large-scale solution to the Internet's problems, but O'Reilly's initiative failed to produce a big payoff. Sales of my Peer-to-Peer book never went high, even though it was recognized in the industry as an indispensable text and was widely cited in both academic and business settings. As I had predicted during our writing phase, no book

produced by other publishers on the subject attracted any attention. We planned a second Peer-to-Peer conference in 2002, but canceled it because the September 11, 2001 attacks by Al Qaeda were depressing travel. I think that an even greater reason was the replacement of the 1990s dot-com boom by a dot-com bust. But peer-to-peer had an ongoing impact in different ways: some on our company and some on the world. I'll use the rest of this section to detail those impacts.

Tim O'Reilly and other managers learned from our peer-to-peer experience that we could be influencers. We had played a historic role years before in promoting the meme of open source, but now we played up our strengths here more pervasively and consciously.

Instead of another Peer-to-Peer conference, we held a series of Emerging Technology conferences. One year, one of the technologies we highlighted was WiFi—yes, it was once new and perplexing! I found these conferences stimulating and accomplished a lot of networking at them, because people still recognized me as a leader and wanted to present their projects to me.

The Emerging Technology conference lasted only a couple years, but it inspired creativity in how we followed new trends. The difficulty with publishing is that very early technologies don't reward major investments—and a book is a major investment by the publisher. You have to release a book at the right stage of the famous Gartner hype cycle: Hopefully you catch something well before it reaches its peak, but when it has more followers than the early adaptors that Tim O'Reilly referred to as "alpha geeks". But an institution like ours, always striving to help innovation along, is critically tasked to hearing what the alpha geeks are doing.

In some sense, our attention to emerging technologies, which we made explicit when covering the peer-to-peer movement of the early 2000s, started much earlier. It was always top of our agendas to be hip to the first flares of new movements in computing. In addition to the brief period during which we ran Peer-to-Peer and Emerging Technology conferences, we used the web site to highlight interesting trends, particularly through the Radar blog. Mike Loukides started a low-budget online journal about biohacking, around the period we were covering health IT. And foremost among our touchpoints with the alpha geeks were our FOO camps, which Sara Winge started and which ventured far outside computing to health, science, education, and policy. The Peer-to-Peer book triggered a turning point in our consciousness.

In my observations, it's fair to credit peer-to-peer with even more grandiose impacts.

»» Although peer-to-peer failed to build a free and egalitarian network, its meme entered the Internet at higher levels. Remember the lesson that peer-to-peer taught: Internet users do not have to be lonely, atomized individuals at the

mercy of central servers. The users can become powerful when they come together. And they can offer their own ideas, not simply be recipients of messages from rich corporations.

In that context, what are the billions of contributions to Twitter, Facebook, and YouTube but the expression of the masses in peer-to-peer communications? During the 1990s, content was controlled by those who ran servers, downgrading almost all computer users to the role of mere consumers. With social media, computer users took the lead and met each other as peers. Creativity broke out of old bounds.

Naturally, all the problems of such connections came in spades as well. Two problems in my 2004 article applied more than ever: The problem of addressing allowed attackers from Russia or Eastern Europe to pose as British or American sources, and the problem of trust became even more obvious.

User-generated content was labeled Web 2.0 by Tim O'Reilly in the early 2000s. Web 2.0 went up the networking stack further than peer-to-peer, suggesting that individual users on their home or business computers would create value on the Internet. (I think Tim himself drew the connection between peer-to-peer and Web 2.0, although I'm not sure.) And Web 2.0 was a challenge to the standard paradigm too—a big one. Superficially, it suggested that the "web portal" concept popularized by AOL and Yahoo!, a model where large centralized corporations offered content to passive viewers, was obsolete or at least as not as fertile as peer-to-peer. Web 2.0 also represented an implicit challenge to the prevailing telecom business model, where cable and telephone companies reserved nearly all their bandwidth to download content to their users, and left users only a sliver of bandwidth with which to make requests for content.

Web 2.0 could theoretically have been implemented through a peer-to-peer network, and there were attempts like Jabber (later standardized as XMPP) to do so. But thanks to the problems I had identified with peer-to-peer networks, Web 2.0 peer-to-peer contributions ended up being implemented by centralized client/server systems. Hence the primacy of YouTube, Facebook, Twitter, and so forth.

By the same reasoning, peer-to-peer informs Government 2.0, a movement I cover in a later chapter. The goal of Government 2.0 was increasing public engagement, a key factor in which was to accept data and opinions from constituents.

Finally, peer-to-peer forced content producers to meet the challenges of the Internet head on, which they had avoided temporarily by eliminating Napster. iTunes for music and Netflix for movies are the studios' responses to the demonstrated desire shown by their customers to receive content cheaply, instantly, and continuously. Jogged by peer-to-peer, the real Internet revolution in media had begun.

I've said that peer-to-peer challenged the hold that client/server technologies had on computing. Under client/server, an administrator (typically at the company where people worked) vetted and installed the software they thought was what people needed. Petitions for new services had to go through a bureaucracy. Technical barriers were used to suppress technical innovation.

In contrast, peer-to-peer offered end-users a way to serve themselves. As soon as they downloaded an app, they could run a service with anyone else who downloaded it. Naturally, security risks swarmed around this freedom. Still, that freedom was appealing to end-users who recognized the value of a service to which their network admins were indifferent.

I remember from this period a Dilbert cartoon where the lead character, the engineer Dilbert, brings home a large device with a screen. He tells his pet Dogbert, "I just bought a videoconferencing system. Now I have to wait for someone else in the world to buy one." In the last column, Dogbert observes, "It's unsettling to realize that progress depends on people like you." Peer-to-peer services embodied just that wild combination of experimentation and trust. But adoption was stymied by another technical barrier: restrictions on the ports used to exchange information.

Readers without much Internet background may balk at trying to get their heads around the technical information they need to understand this social phenomenon. But hold steady and read on. Once again, we have an excellent lesson in the impacts of obscure technical matters on society, and on the changes in attitudes that must take place to enable technological change.

Each computer system is guaranteed to receive the traffic sent to it because the computer has one or more unique Internet addresses. But the traffic can be an unruly mix of packets sent by different applications—email, web, network administrative tasks, and so on—all jostling each other as they arrive.

Therefore, Internet software assigns each application an arbitrary number known as a "port", mimicking the physical ports into which electricians plug cables. Email is assigned port 25, the Web gets port 80, etc. A special non-profit organization, the Internet Assigned Numbers Authority, assigns the most important numbers. Other applications can just pick some arbitrary high number and use it consistently.

The concept of ports presumes clean divisions between applications. You would no more expect email to come over the web than you would expect a dog to give birth to a parrot. The sanctity of the port numbers is paramount.

The early peer-to-peer applications picked their own numbers, but quickly found that they couldn't get through corporate firewalls—the software and hardware responsible for both aggregating and filtering out traffic. Here again, administrators were policing computer use by restricting access to traffic on just a few ports. The administrators presented this as a security measure, but its main effect was to prevent users from running the applications they wanted. Such screening could also be problematic for the classic File Transfer Protocol, which used random high numbers.

Blocked by standard firewall rules, the peer-to-peer developers gazed yearningly at port 80. It was reserved for Web traffic, but the popularity of Web ensured that this port had to be open on every computer used by typical computer users. And so the peer-to-peer developers made a momentous decision: They would violate the tradition of providing a unique port number for their application, and send everything over the Web. The user's web browser would receive and process the traffic.

Traditional network experts were appalled at what peer-to-peer developers were doing. The traditionalists called it "port 80 pollution". (We encountered these hecklers in an earlier chapter, complaining about browsers that opened multiple parallel connections to download images.) But it led to a practice so universal that we all use it all the time—and yes, we receive our email over the Web. We also use nearly every service provided by every modern software company, and we call it web services. Port 80 pollution has become Software as a Service (SaaS).

Although web services are unlikely to be supplanted, Web. 2.0 is dangerously at risk, thanks to the problems with identity and trust that rendered peer-to-peer mostly unfeasible. The whole Web 2.0 movement depended on Section 230. This term, a rallying cry on the Internet, refers to the one section in the Communications Decency Act of 1996

that survived its Supreme Court challenge. The law protected web sites such as YouTube and Facebook, saying they couldn't be found liable for content uploaded to them by individuals (so long as it was identified as the individuals' content, not owned by the web site).

Ironically, one of the first cases to invoke section 230 was ruled incorrectly. Matt Drudge had contracts with America Online to publish his right-wing political content, often with a prurient and sleazy twist. (Drudge did act legally when he broke the Monica Lewinsky scandal that brought Bill Clinton to impeachment.) In 1998, Drudge posted fallacious material about a political operative named Sidney Blumenthal, who sued both Drudge and America Online. The court exempted America Online from paying out a major settlement, which I consider an absurd interpretation of Section 230. America Online recruited Drudge and paid for his content, and therefore should have been slapped with responsibility for his lies, according to any responsible interpretation of the law.

But Section 230 was generally a good law, allowing Web 2.0 to take off and sites such as YouTube to make real contributions to education and culture. Of course, these sites became extremely exploitative of contributors, visitors, advertisers, and news sites in many ways, but one can't blame the business model on Web 2.0 or Section 230. These sites also invest a lot of money taking down objectionable and illegal content, partly to meet the demands of Section 230 and partly to assuage public opinion. Unfortunately, they are currently in a losing battle with the trolls.

I still have hope for the Internet. But it was obvious by 2015 or so that user-uploaded content was creating real problems. Governments were already trying to weaken Section 230 (which was imitated by many other countries, although with some important differences). Sometimes the governments' stated concerns about porn, terrorist content, and other bad actors were veiled attempts to curtail freedom of speech, but there was real meat to their complaints as well. And limitations on content, by both private sites and government regulation, are sure to grow. It's all a response to Web 2.0.

Anticipation and realization: Linux (late 1990s)

What big movement carried along my work before peer-to-peer? Nothing else in my career (or most careers) quite corresponds to that heady combination of writing opportunities, speaking engagements, technical inquiry, and social significance. But another fulfilling time at O'Reilly was the period of the late 1990s, when we threw in our reputation and our future with free

software—open source, if you prefer. This movement, which gets a chapter of its own, shifted the whole computer field onto a new track, and along with it many processes in the larger society. O'Reilly drove the movement forward with a couple high-sales series. The chief series was on the Linux kernel, but MySQL was quite important too. I edited almost all the books in both series.

O'Reilly made history in 1998 through a summit whose attendees agreed to promote the term "open source" as a more easily understood moniker for free software. (I wasn't at the meeting.) We performed a similar word twist on our Perl conference, renaming it the Open Source Convention. We had also put stakes in the free software terrain by publishing Eric S. Raymond's book *The Cathedral & the Bazaar* in 1999. But it took a while for O'Reilly managers to convert abstract advocacy into business opportunity; to realize the bounty that free software offered us.

Why have I titled this section after Linux? Because during the late 1990s, people who heard of free and open source software associated it first and foremost with the Linux kernel and surrounding operating system—that's how important Linux had become.

I had heard of Linux quite early, within a few months of its initial release by Linus Torvalds in 1991, while I was still employed at Hitachi. Still, when my friend Lar Kaufman explained it to me, I had scant interest in the new operating system. I told him, "There have been many Unix systems, some ported to personal computers. What's special here?" Hitachi, where we both worked, had shimmied up a rotting branch of the Unix ecosystem called the Digital Computing Environment (DCE). Unix still dominated servers and advanced computing, but was reaching the end of the road.

My first privilege to touch Linux came in early 1992 when Kaufman brought me to a friend's house, where we gently inserted and removed, in order, the 51 or 52 floppy disks provided by Softlanding Linux System. (By the way, it would be a few more years before enthusiasts of the free GNU tools started to attach "GNU" before "Linux".) SLS was not the first commercial distribution of Linux, but it was the first that people found robust enough to take seriously. SLS provided the basics for Linux, including graphics. You couldn't complain. That market soon became quite crowded, attracting the attention of software distributor Bob Young and prompting his own Linux distribution through the new company Red Hat.

After I saw Linux overtaking the field and crowding out free alternatives such as FreeBSD, I became an evangelist for it at O'Reilly and drove a book series that became perhaps our most noted and respected offering in the late 1990s. Linux was becoming quite the rage, standing in for everything happening in free software.

My shift of focus, from the quasi-standards that failed at Hitachi to the overboiling creativity of the Linux community, took place through random chemical bonding instead of deliberate strategizing. My journey mirrored the upheaval in the computer field as a whole. Look at what happened at professional computing companies such as IBM and Hewlett Packard: Like Hitachi, they all lined up behind DCE in the early 1990s and poured resources into making it the center of their offerings, but by the late 1990s they were all vying to offer the best Linux support.

DCE was unviable from its very premises. It was an overly ambitious attempt to tie computers together from different manufacturers, using the software considered best of class from each manufacturer. The whole thing was coordinated by an understaffed and underfunded organization the manufacturers threw together and called the Open Software Foundation.

OSF could not fulfill its role as a standards organization. It was a marriage of convenience between computer vendors who harbored no fidelity to one another. All of its offerings came from member companies, loaded with bugs and thoughtless design decisions (known to programmers as "technical debt") that reflected short-term advantages at the time of their development years before. Taking poor quality components, and then piling on the additional impossible challenge of getting them all to cohere harmoniously, was a travesty of a project. It wasted the efforts of thousands of highly paid developers—not to mention earnest tech writers such as my team at Hitachi.

A deeper analysis provides a shorthand to sum up the previous complaint: There was no community around OSF. There was only a gnarly tangle of idealistic aspirations dragged down by grubby corporate considerations. OSF was last seen in an announcement that they had merged with some other obscure quasi-standards organization, thus reducing by one the immense number of obscure quasi-standards organizations worth ignoring. Many years later, the initials must have been considered free for the taking, because the OpenStack Foundation started using them.

Linux, by contrast, formed a community early and built everything in a truly open fashion. Torvalds's adoption of a free license, which he often said was the best decision he made in his career, was just the starting point. Thanks to the license, a true community could build up around the software, keeping everybody honest and allowing for constant injections of new energy. As one example of the community's strength, Linux soon ran on more hardware and supported more devices than any other operating system in history. No less important was the zealous love pledged to Linux by people around the world, many of them unschooled in the technical details that made it work.

So I had matured into a firm backer of Linux by the time I came to O'Reilly. Even though Linux was confined to desktops and small servers, hardly noted by a computer industry consumed by the battle between Microsoft systems and Unix, O'Reilly entrusted me with developing a series I took on this task methodically and with high-quality results.

Kaufman also kept up his interest in Linux and fed us valuable advice at O'Reilly, where I am fairly certain he took a job for a while before he went to law school. Kaufman even suggested the Wild West theme that our brand expert Edie Freedman adopted. Kaufman offered us a book with beautiful woodcuts of the nineteenth-century American West, which became the first pictures we used to illustrate our Linux books. In fact, our artist would seek out a unique picture to start each chapter of each book. After a few books, this became prohibitively time-consuming and we stuck to one image per book.

But in 1994 or 1995, Linux had not yet entered major data centers and become almost indispensable for large servers or virtual computing. It certainly didn't turn up on cell phones and other mobile devices. Developers hadn't noticed Linux's value for offering graphics and robust networking on embedded systems. Our books sold rather poorly, so we gave up on Linux.

When O'Reilly managers told me to stop looking for Linux-related topics, I did not push back. I could have carved out time —as I was always doing to indulge my own intellectual interests—to stay in touch with the Linux community, but I didn't. I think the reason that both I and my managers turned our backs on this phenomenon was our lack of understanding concerning the historic social change of which Linux was both impetus and beneficiary. It wasn't just we who missed it—the whole world was slow to catch up. In particular, sociologists and economists on the whole had no clue what Linux represented, and they were just starting to nose around free software in the mid-1990s.

Now we have hundreds of scholarly, insightful explorations along the lines of Yochai Benkler's *Wealth of Networks*. So now, standing on the towers of books and research papers (including a couple by me) that have appeared on free software and open source, I can analyze retrospectively what happened.

I won't bother here with an analysis of the factors in the mid-1990s—globalization, the technical and social state of the Internet, and technical optimizations—that drove Linux to world domination; such punditry has been exercised by better observers. The Linux community has never been perfect, of course. Its problems with negative interactions, coming down particularly hard on women, aroused widespread criticism. After free software communities came to a general understanding of the traits needed to

maintain participation, the cultural shift overtook Torvalds himself and led him to humbly relinquish his leadership position in Linux.

But in the meantime, Linux gave the free software movement both a proud platform for limitless innovation and a success story to shout from the rooftops worldwide. It inspired lawyers, economists, policy-makers, educators, and many others including (yes) publishers to re-examine their assumptions about ownership, value, and transparency, leading to a plethora of "open this" and "open that". Linux also cemented my own commitment to free software, while offering me a path to a strong career as editor, writer, and advocate.

But the payoff didn't come early enough to save my Linux work at O'Reilly. During the mid-1990s, managers tossed a diverse group of wan, uninspiring topics at me. I edited a book about virtual private networks (VPNs), a topic of continuing importance. VPNs are marketed to all network users from the home to the largest corporation. But VPNs at that time were mainly proprietary. Because of that, I think, the book didn't sell. I edited some Microsoft-related books too, and those didn't sell either.

Finally, later in the decade, we hired an experienced editor named Mark Stone, who possessed a strong understanding of technology and had watched the growth of Linux with excitement all the time that attention toward it was in remission at O'Reilly. Stone taught all of us a lot about what was going on in the Linux area. Over just a couple years, Linux had matured and taken on worldwide significance. He inspired me to pick up work again on Linux and other free software. We had no time to lose. As one of our editors, Paula Ferguson, said at the time, "O'Reilly is often either too early or too late to jump on a technology—or both, as in the case of Linux."

The following six months were frantic. I lined up the original or new authors to update all the existing books in our Linux series, and solicited new books as well. The effort paid off handsomely, and sales were gratifying up to the early 2000s. They suddenly declined then, as so many of our series did, for reasons I have already cited, related to changes in the publishing industry and where programmers directed their attention.

But the shift signaled by our re-embrace of Linux went far deeper than a business decision to support a book series. It placed us back into the center of the free and open source software movements. We renewed our vows with these movements, which had been formed long ago by our coverage of Unix and the X Window System.

Many of the leaders in the free and open source software movements collaborated with us closely and even engaged in strategy with us; I became their friend, ally, and supporter. I jumped eagerly into the role of public advocate for free software and all the social changes it brought in its wake: free culture, open government, and more. My advocacy and writing abilities brought me to the Googleplex in Silicon Valley, to Brazil, Brussels, and Amsterdam, and to all places online where free software was under discussion. I was now part of a global community that was simultaneously idealistic and pragmatic, a Commons where work and personal connections intersected, all fueled by the passion for a new way of doing things.

This was the last glow of computer books' golden age, which lasted a couple decades. The era started with universal adoption of personal computers in the 1980s and intensified as Internet access in the 1990s made those computers supremely valuable, then trailed off as more and more information came online for free. In the 1990s, people just automatically bought a computer book in their quest for needed skills. From the late 1990s onward, and particularly after the dot-com bust of 2001, people turned first to online searches and had to hit a barrier there before reaching a rational conclusion to buy a book. Switching gears mentally from "find it for free" to "pay for it" was a big hurdle, even though most people could easily afford a book.

I always maintained a healthy skepticism of the dot-com boom. Those of us with knowledge but not cash held back from overexuberance. Those with cash and little knowledge invested billions and lost them. I remember how a customer saw the word Linux on a T-shirt I was wearing one day in a convenience store during the 1990s. The customer said, "Gee, I've been thinking of investing in some of those new computer companies." I told him, "Don't invest—instead, study the technologies, so you can become an expert user." I hope he took my advice.

A ludicrous controversy: Smileys (1997)

Until the funk that descended on me in the mid-1990s, there was only one time I feared getting fired. It was over a subject so trivial I'm embarrassed to air it now: a book about smileys. Yes, those silly little graphics that indicate emotion or convey an ironic message that's in tension with the overt statement in the text.

Smileys are now pixel graphics, supported by most tools on a fully graphic monitor. Although most people don't realize it, smileys are carved into stone—or actually the closest thing to that nowadays, which is to be included in the Unicode standard for characters and symbols.

Back before all these graphical representations, smileys were clever manipulations of plain Ascii characters, starting with the original :-) for a smile. Over time, clever inventors people published hundreds of these text representations of everyday gestures. Dale Dougherty thought it would be fun to put out a book containing all the smileys we could find online.

As the book went through production, grumbling from the production staff suffused through our Sherman Street office. Many of the smileys, they said, were offensive. They would not perhaps be considered obscene, but they contained demeaning stereotypes of women or minorities. Living in Cambridge, the staff were alert to these risks.

Dougherty would have none of it. He agreed that some smileys fell outside of good taste and might be offensive, but he seemed to think that it was our role to present the Internet authentically. Here I'm straining to be fair to Dougherty and to present an argument that seems reasonable, whether or not he would present it that way. One could argue that the full collection of smileys was required to convey the richness and diversity of the Internet, including aspects that might make enlightened readers uncomfortable.

The in-house controversy over this silly little project, a dumb distraction from the serious work of teaching professionals how to administer and program their systems, was rending the company. I finally decided to look at the draft myself. It convinced me that Dougherty was wrong and that we needed to follow the judgment of the protesting production staff. Sample smileys included "Hottentot" (with an exquisitely text-fashioned bone through the nose) and "Mexican run over by a train" (a sombrero atop two parallel lines representing tracks). I was afraid that O'Reilly's reputation would tumble into the gutter by releasing a book with such disgusting stereotypes.

So I used the medium of email—not the first time I have been an internal activist in a company over the email medium—writing to Dougherty, Tim O'Reilly, and the other editors to insist that we heed the production staff and take out offensive smileys. I was terrified by challenging Dougherty, cofounder of the company, directly. I was ready to be fired.

I waited. Discussion ensued. Mike Loukides, who has always held a leading role and been highly respected at the company, came out in our support. Finally, Tim himself called me at home to say that he agreed with me and the production staff, and that the sexist and racist smileys would be removed.

Tim's executive decision was a welcome salve to a company-wide crisis. But it was also a moral choice in the Titanic battle between two philosophies: one calling for a detached observation of life as it is, the other reflecting a vision of a better life. I think the

results of this choice can be seen in many other positions taken by the company in the decades since.

☞ *Intellectual prosperity: Writing and editing*

Intellectual prosperity: Writing and editing

Contents of this chapter

Sinking books (late 2010s)

Before entering the computer field, I spent a year trying to earn a social work degree. While counseling authors at O'Reilly, I often felt I had opened a wormhole and transported myself back to that social work program. I dealt with the pressures of authors' work, pressures of their loss of work, conflicts with urgent family needs or other time commitments, struggles with their depression, and everything else that comes into the life of a human being.

Such patience proved valuable during my later years, when O'Reilly managers often plunked me down into projects that I had no part in planning. Many of these projects had lacked oversight earlier, as they crept their way to a dead end. I was often asked to fix a mess, sometimes at breakneck speed, and without much guidance.

One sad book project brought in a couple networking experts who had created quite successful presentations on that subject and had been recruited by O'Reilly to do a book. It quickly became obvious that they fell abjectly short of fashioning a coherent prose narrative. But no one at O'Reilly had a longitudinal view of their work—a problem I could never imagine happening during my first 10 or 15 years at the company. Instead, junior editors were assigned at various meandering stages of the authors' writing to suggest changes. Each editor recognized and tried to react to the multifaceted problems, but with a different perspective, using different words. The authors couldn't make progress because they kept getting hit with dire assessments, always shifting.

Somehow, these poor fellows produced a complete draft. It even went through a tech review that highlighted some of the same organizational problems that the editors had clumsily tried to address. So numerous people—not just the authors and O'Reilly staff—had been inconvenienced by this misdirected project.

To render the situation even thornier, a sponsoring company had read early versions of the muddle and found useful material in it. It continually amazes me how people can approve patently unviable writing. So O'Reilly had already made money licensing the early versions of some chapters, making it even harder to end the project.

Trapped in a situation with no redemption, I treated the authors with respect and engaged with an intensity that no one had granted them previously. After discovering that they could not place their ideas into a sequence that readers could follow, I started writing whole chapters by myself. This is something I had never done for an author before, and I knew it was ridiculous but felt I had to do whatever I could for them and for O'Reilly.

Luckily, I was rescued by my boss at that time, Rachel Roumeliotis. Roumeliotis had not been at O'Reilly for long—at least, compared to the people with whom I'd worked at the beginning of my career there, whose tenures spanned geological epochs—but she rose remarkably fast in the organization. It seemed like she was always taking on more and more. For instance, she planned several conferences and started new ones in addition to all the book work. (This made sense in our integrated media strategy, where all our activities intertwined.) She was much younger than me, but happily accepted a stack of old 12-inch LPs of classic 1960s rock music from me when I was trying to clear out my basement.

So Roumeliotis laid down the law: It was not my job to write a book for the authors, and if they could not complete the job in a satisfactory manner, the project would come to an end. Although the authors expressed understandable frustration at the

termination of a project they had been permitted and even encouraged to drag out for so long, they remained open to doing other kinds of material for the company. I was thankfully relieved of further responsibility.

Another book about the same time provided a happier outcome. This book was also on the cusp of cancellation after having been denied expert attention. The half dozen chapters produced so far on data management lacked discipline and direction, but I sensed some nuggets of value that steeled me to save the book.

Unlike the authors on the project previously described, this author could listen to editorial advice and pull together his work. We worked together without blame and without resistance to identify the concepts that required emphasis and wormed out the valuable material from his edifice of verbiage. As is typical of disorganized books, this one was full of redundancy. Reducing the material to essentials was highly gratifying. The book was ultimately finished and released to some acclaim, along with enthusiastic support from a sponsor.

Throughout my career at O'Reilly, I handled both projects that sprang from my own convictions and projects handed to me by others. Both categories included great successes and embarrassing failures. The best stories from these writing projects form the subject of this chapter.

Spanning the globe: Regional reports (2016)

Not all my projects at O'Reilly concerned mind-numbing technical issues such as data storage—although I always tried to bring alive even the most mundane topic. I landed a few lavish opportunities for creativity, including assessments of computer tech opportunities in different cities or countries in 2016.

Roumeliotis told me the company was considering a series of reports on promising tech regions, a bid to get attention from people living in those regions and increase our followers and conference attendees. I went through some twenty years of files on authors and tech reviewers to find people who made their careers in places as diverse as Ottawa and Lawrence, Kansas. I turned up lots of detail through my interviews and stored them all in files that I turned over to Roumeliotis. None of it got used, but she did give me a few assignments.

Roumeliotis thought regularly about appealing to new demographics. This was around the time she moved our Open Source Convention from Portland, Oregon to Austin, Texas. She thought maybe we had maxed out our exposure in the Pacific Northwest and would attract a lot of new people from the Midwest by relocating to the middle of the continent. Our first year in Austin (which I attended) drew a pretty good crowd, but in the second year (which I didn't attend—the first Open Source Convention I ever missed) the draw wasn't gratifying. Roumeliotis took the convention back to Portland.

My part of the strategy, which involved writing reports to drum up business in new areas, also failed. But it was very rewarding for me, and I went much deeper into the subject matter than I think anyone expected. It provided me a cultural odyssey.

First I was asked to write about London, where the financial ("fintech") industry anchored a strong computer sector. I drummed up interviews from a number of contacts, and learned about interesting differences between the English and American views of business. Essentially, U.S. venture capital is distorted by a "unicorn" model, hoping that every company will become a billion-dollar firm in a few years. The English are more patient, looking for gradual, organic growth. This and many other insights went into my 16-page report.

By extraordinary (in all three English syllables) luck, I had planned a personal trip to London with my wife during the time I was writing the report. All sorts of local references and joking puns about London life made it into the report, along with insights about the difficulties of living there. I held a meeting in a pub with some of the people I had interviewed for the report, which led to some networking that was gratifying to see. I also took some pictures of 3D-printed objects in the Victoria & Albert Museum to flesh out a short section of the report on technology and art; I had tried unsuccessfully to get interviews with Londoners in that area.

I put even more of myself into a report on Brazil. Roumeliotis had asked me to cover open source throughout Latin America, but after several interviews I determined that there was little to say about it outside of Brazil. As the largest Latin American country, and one that has cultivated its tech sector diligently over several decades, Brazil has become a center for free software development. It's no coincidence that I had lots of contacts there. I also got a feel for its free software scene after my attendance at a FISL conference and my work with Marcelo Marques on the Hackerteen comic book I'll describe later. I had picked up a bit of Portuguese to prepare for my visits. So I persuaded Roumeliotis that the report would be much more useful and focused if I stuck to Brazil.

I worked hard to find out what was going on with free software since my previous visits, and to be inclusive of marginalized populations, particularly women and small towns outside the major metropolitan areas. I got wonderful interviews and very

dedicated reviewers, including my Hackerteen friends and Jon "maddog" Hall, a former technologist at Digital Equipment Corporation who made it a personal mission to promote education and free software in Brazil. I also worked into the report references to relevant books and movies.

In a coup de grâce, I decided to start each section of my 19-page report with a pertinent line from a famous Brazilian pop song. I had always enjoyed the pounding Samba music and the more polished bossa nova, and I knew the names of the most famous Brazilian songwriters. Internet searches readily turned up a huge range of songs, and between my minimal knowledge of Portuguese and some dictionary work, I picked an apt citation for each section.

It was these cultural references to pop songs that won over a Brazilian translator who was recommended by one of my reviewers. I got O'Reilly to approve a thousand dollars to pay for her translation, all in pursuit of getting our name in front of more Brazilian professionals.

All these reports called for a stretch in expertise that many writers would eschew. Caution is one of the chief enemies of strong writing. (Tact is another, as I mention elsewhere.) To compensate for my lack of caution and tact, I do a lot of research. I get reviews from authorities in the field, whose email addresses I hoard like precious stones. But most of all, I create a compelling argument that gives urgency to my claims. Lack of caution and tact doesn't force a writer to adopt extreme positions or ignore counterbalancing arguments. As my Platform Independent columns illustrated, my writing tried to be radically and fervently level-headed.

As said before, the London and Brazilian efforts didn't seem to pay off. I was put on one final regional project: an overview of the Austin tech scene. I found lots of fascinating historical background, and some disturbing evidence that Austin was starting to suffer from the same economic polarization and social disintegration as San Francisco, Boston, and other tech-friendly cities (London among them) caused as affluent computer people took over neighborhoods and raised rents beyond what other residents could afford. My research brought some valuable contacts that could have assisted us in further networking.

I proudly turned this brief report over to Roumeliotis, who informed me that it didn't meet O'Reilly's PR goals and therefore would not be published. So I put it up on the popular Medium blog site. The company's concern with Austin ended when we moved our conferences out.

A golden iron age: Manufacturing and Solid (mid 2000s)

Another commendable but ultimately unsuccessful O'Reilly venture offered me a novel chance to contribute to a new field —but also a hefty dose of frustration.

Our management saw the effects that digital technologies were having on manufacturing and product design. Companies in these spaces are normally less tuned-in to the potential of data analysis than industries such as finance and defense that have to keep a fine competitive edge to survive. But with the rise of machine learning and the new streams of data afforded by the Internet of Things, manufacturers were starting to see a benefit to digitizing old processes, and O'Reilly wanted to be there to push the transformation.

As with most new initiatives at O'Reilly, the main offering was a conference. I already saw this with health IT and web performance. Conferences help to create community and raise awareness, much more than a book, and if they're successful they rake in a lot more money than books. This is the central tragedy behind the ultimate destruction of the conference group described in the first section of this memoir. Not only did conferences bring in revenue; they lay at the core of O'Reilly's strategy for exerting influence.

Analytics in manufacturing were championed at O'Reilly's Solid conferences in 2014 and 2015. Demos ranged from enhanced CAD tools to augmented reality. Keynoters laid out glowing visions of the future of manufacturing. I interviewed people from different continents about their innovative approaches to product development.

The venue was a major part of the fun: an old set of industrial buildings (actually a converted military base) called Fort Mason, near the Marina on the north side of San Francisco. The city had refashioned these buildings into a viable conference space, and the sense of connection to a blue-collar past added a sense of reverence to attendance.

Manufacturing is considered a "vertical" in business jargon. It's hard for publishers to make money on verticals. A conference on data in general, or a book on networks in general, will generally bring in more revenue than one focused on data in manufacturing or networks in manufacturing. After two years, Solid was discontinued.

But along the way, I got a cool gig for a few months. Somebody in marketing contacted me in March 2015 to say O'Reilly had promised an article to a web site run by the International Manufacturing Technology Show. The deadline was alarming—they wanted me to submit four or five hundred words in about 48 hours. Moreover, our marketing person could not provide any of the answers to the questions that a writer needs to ask, such as who the audience was or what IMTS was looking for. I had never heard of them, but I thought up a creative idea about the four criteria for judging the value of data (integrity, timeliness, and so forth) and delivered a pretty decent piece. Although I didn't get any feedback from O'Reilly or IMTS, they gave me the impression that they were satisfied. I did not judge the marketing person for the slipshod manner in which I had been approached. Not knowing the background, I ignored the signals of a poorly managed project.

A month later, another email from the same marketing person. "It is almost time for our next submission to IMTS..." I realized that the commitment she was expecting from me was not a one-off piece, but a monthly series. She had just done exactly what she had done the month before, hitting me broadside with a last-minute request.

So this time I tried to inject a little discipline into the commission. I insisted on a phone call where marketing would lay out what they wanted and some basic parameters about the articles I was going to produce. I never got any guidance, but I quickly threw together another acceptable article (this one about robots in manufacturing) and settled down to plan my series.

Believe it or not, I was having a really good time with the IMTS articles. I found a couple blog sites about manufacturing, which gave me some interesting background and suggested to me that my subjects of interest were not being covered online. I wrote about supply chains, predicting monitoring, and ecological concerns.

I never heard a peep from anyone at IMTS; I simply tossed my articles over a virtual wall to the marketing person at O'Reilly. A few months later, though, I noticed that she was not responding. Repeated email went unanswered.

A few more months passed. Suddenly I heard from a new marketing person. The former one I'd been dealing with had left under a cloud—I suppose her managers noticed the same behavior on other projects that I had suffered from on the IMTS articles. The new marketing person apologized for ignoring my last few email messages. She had no idea O'Reilly was working with IMTS and couldn't fathom why I was sending articles. The transition was obviously handled with the same conscientiousness as the writing project itself.

Now I had an ally again within O'Reilly. But a new barrier presented itself: Our marketing contact at IMTS had also left the company, and no one was picking up the articles I had sent. My marketing person assured me that someone new would be hired and that IMTS would resume publication. That was the last I ever heard about the project. I didn't even get the dignity of a formal ending. The bitter taste left by this project, unfortunately, was a familiar one. We'll soon see another version of the same script in play.

Still, I had a good time covering manufacturing for a few months. The project allowed me to explore new areas for the application of computer technology. And it allowed at the least the illusory feeling that I was doing something valuable for both companies and their readers. (My articles can no longer be retrieved from the IMTS site.)

January 2023: Having some free time after wrapping up a major contract, I decided to clean up some old files. I discovered drafts of nine articles I wrote for IMTS, and reposted them to my personal web site.

The digital gets messy: Health IT (late 2000s—2010s)

In the cozy field of health IT, many people recall O'Reilly's involvement with pleasure. But the initiative has been almost totally forgotten outside this group. At its peak, the health IT effort at O'Reilly encompassed an annual conference focused on this subject as well as a track at our flagship Open Source Convention, a book series, an open journal on biohacking, and a steady stream of blog postings and video interviews that I mostly provided. But the company eventually found that they could not make enough money in this area. Like manufacturing, health care turned out to be an underperforming "vertical".

I came to health IT out of the same curiosity that led me to explore Linux or peer-to-peer. I can trace my first health IT blog posting to 2003, when I covered the company Vocera as an interesting combination of database technology, 802.11 networking, and voice recognition. I recognized health IT as a special segment of interest around 2009, which is also when it came to the attention of O'Reilly management.

My initial advice to the O'Reilly managers and marketing team involved a lot of reorientation: recognizing the abbreviation CMS as Centers for Medicare & Medicaid Services instead of Customer Management System, and refraining from saying that health care is waiting for its killer app. Because health care institutions in the Obama administration were among the first and most sophisticated proponents of open government, Tim O'Reilly was familiar with many health care reformers as well. One of his

particularly notable and valuable connections was Todd Park, who moved from the position of CTO in the Department of Health and Human Services to being CTO of the federal government as a whole. In short, Tim and I were on parallel tracks in the health care space, investigating the potential for our company's health IT work and serving as advocates for technological change.

Because O'Reilly always joined a community and entered a technological area by holding a conference, management decided to extend the successful concept of a data conference to health IT. Because they were running several conferences called Strata, they started a narrowly focused one named Strata Rx. It was a critical, but not financial, success. I was told that attendees were very excited by the content, but spent their free time making connections in the hallways and mostly ignored the vendors. Of course, vendors can't justify coming to a conference where they spend their time staring blankly into the busy hallway. So we didn't have the money to continue.

The same goes for the three books I edited on health IT: well-received, and even regarded as indispensable contributions to the field, but not actually bought that often.

The collapse of our health IT venture came in early 2014 just when I was putting my final touches on my flagship work in this area: a comprehensive report covering everything in the health IT space from medical records to devices to research. At 50 pages, it's the longest single project I ever did at O'Reilly.

My only disappointment with the report itself was a complete lack of graphics. I asked numerous contacts for photographs of them or their colleagues at work, but the notorious reclusiveness of the health care was interposed. Most recipients of my request cited some kind of need for confidentiality. I did receive one photo, but it was lousy. The production staff at O'Reilly tried to be helpful, giving me access to an enormous archive of stock photos they had licensed. But all the photos related to health care were glossy marketing images of clean, young, attractive people smiling over their computers or in their examination rooms—completely the opposite of the message in my document, which faced difficulties and disparities in the health care field head on. So I finished the report with no images. There is perhaps some bigger message in my coming up empty when seeking images that reflected the reality of health care and the role of IT in it—the field might not have been ready for the transformation I was proposing.

The report had been blessed by management a few months earlier, and I turned it over to production with the expectation that we'd use it to give a boost to the health IT effort. I don't think I knew that the end was near.

During the brief time the book was in production, the company decided to forgo further Strata Rx conferences and to dismantle the Strata Rx team. Julie Steele, the editor who had run Strata Rx, took a job with another company. By the time production finished my report and released it, not only did the company totally ignore it—The company lacked any staff or processes that could have given it attention.

Thus, although the report was officially released by O'Reilly and put up for download, no publicity went out—no press release, no tweets on social media, nothing. I wrote directly to Tim O'Reilly, figuring that between his long interest in health IT and his friendship for me, at least he'd use his powerful social media presence to mention it. But he didn't.

I was prepared to exploit my own wealth of connections and channels in health IT. I painstakingly made a list of the hundreds of people whom I'd talked to about health care over the years, and sent them copies of the report. I carried out a personal media campaign, contacting media outlets throughout the country and major medical institutions. A handful of people promised to read the report, including some who could have been influential, but I never heard back from the most important ones. The feedback I got from people who did read the report was that it was accurate and valuable, but too long at 50 pages for most people to invest time in.

And thus the health IT effort at O'Reilly came to a disgraceful end. I kept blogging for other publications, and even spun my report into a 45-minute presentation that I delivered twice, thanks to my contacts (once at the nursing school at the University of Massachusetts in Boston, and once at a health IT firm). Occasional later efforts by other O'Reilly editors have produced scattered articles and other works of interest in health IT. But O'Reilly wasn't alone in finding disappointment in the health care space. The whole promise of the 2009 bill that kicked off a great reform effort in health care (unconnected to and long preceding the Affordable Care Act), remained unfulfilled at the time I write today. The willful refusal to coordinate care and share data has probably contributed to many deaths during the COVID-19 period, while the nearly instantaneous adoption of telemedicine to cope with physical distancing shows how the field could have adopted beneficial reforms earlier, had practitioners cared.

Cartoon characters: Hackerteen (late 2000s)

We've seen, during the course of this memoir, that O'Reilly has passed through periods of intrepid experimentation. Someone who came up with a well-considered proposal that could open up new possibilities, no matter how unconventional the

proposal, could get the resources to try it. I myself have pushed the limits of O'Reilly's publication strategy a few times.

The most way-out project I ever did was a comic book called Hackerteen. The book's origin was as strange as its concept: I found the author on the show floor of a Brazilian free software conference.

Here's how I ended up there: One of my authors was feeling guilty because he wasn't meeting his deadlines. Being active in the Brazilian free software community, he made amends by getting me a speaking gig at a major free software conference called FISL.

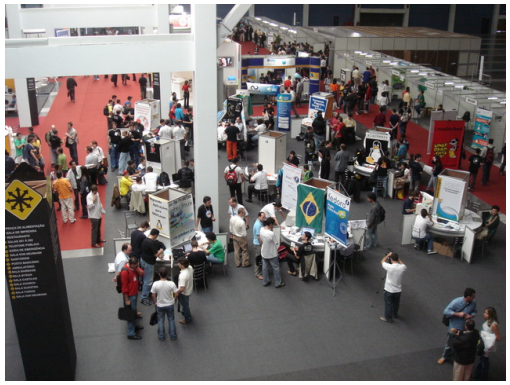
Picture of me with Bruno Gomes Pessanha, the author who brought me to FISL

A tight-knit group of North Americans and Europeans came for the conference to Porto Alegre, described by "maddog" Hall as a significant technology center. We were all housed together in a hotel. I knew a good number of these free software programmers and advocates, so it felt like summer camp to be with them. Furthermore, many attendees, including maddog (as he prefers to be called), attended the conference year after year. maddog was one of the first North Americans to discover a strong free software movement in Brazil, extending to the highest reaches of the leftist Partido dos Trabalhadores that ran the government for many years. He came to Brazil regularly, where he helped erect training and advocacy groups.

My talk was one of a series that I had given in various places on community documentation, the term I created for volunteer-produced manuals and other help in free software communities. This subject got me a lot of traction. I delivered talks on community documentation in the O'Reilly booth on the show floor of the Open Source Convention, then at a formal session at that convention, at a Google facility in Tel Aviv and an Intel facility in Jerusalem during my first trip to Israel, and now at FISL.

The conference organizers tried to plan everything we needed, including a bus from our hotel to the conference, but time and chance happens to them all, so our bus was unreliable. My own talk was the first of the day, and the bus brought me to it late. Very few were in the audience, but otherwise the talk went well enough. On another day, with no bus on the horizon, four of us from the hotel agreed to share a cab to the conference. Richard Stallman was one of the crew, and I got a rare chance to hear his reasoning, as I discuss elsewhere in this memoir.

I visited all the booths on the FISL show floor and made some valuable connections, including a meeting with a Brazilian publisher who wanted to translate some O'Reilly books. But the most fruitful introduction was to Marcelo Marques, the heading of a training company called 4Linux. I may have been introduced to Marques by the author who brought me to FISL, or may have just struck up a conversation because he was a very sociable guy. As he described the education 4Linux offered to young Brazilians who hoped to forge successful software careers in free software, I noticed some interesting cartoon panels on the walls of his booth. The panels had striking young figures in bright colors, and I saw intriguing bits of storyline in the Portuguese speech bubbles.



Marques explained that he had engaged an artist to make the cartoon panels as a catchy marketing campaign. There was no actual comic strip or comic book, but I encouraged him to create one. I thought this would be a wonderful new departure for O'Reilly.

From time to time, editors at O'Reilly would talk of the vast, promising market represented by children and teens interested in computing. Rather like tobacco and soft drink companies—although with positive social values—we wanted to capture our audience while they were young. Nothing ever came of this perennial discussion, but the book I worked out with Marques came closest. O'Reilly managers agreed with us to try the comic book as an educational prop.

Marques and I were strongly aligned on the concept. We wanted a thriller with drama and strong, positive teen characters. But there would be clear messages as well: Hacking is a positive activity, but can be abused. Good guys can use their hacking skills to protect people from the bad guys. Our concept for Hackerteen beautifully reflected my views and values about computing, policy, and free culture.

This is what I pitched when I returned home, and my managers got on board. Marques wrote the text with a colleague, Rodolfo Gobi, and hired an artist with a strong Manga influence and a Brazilian sense of dramatic color. I made a lot of edits. In particular, I chastised the authors for having an all-male team of hackers—and possibly they were all light-skinned as well. We added a girl and a dark-skinned character.

I also carefully checked the technical content, which dealt with intruders and an attempt to bring down the Domain Name System, to make sure it was accurate. I missed just one point: It was unclear how the bad guys had gotten access to the camera on one character's computer in order to capture pictures of her naked and blackmail her. The mechanism was never explained, and turned out to be weak when I later asked Marques about it. Still the book was quite good both technically and as a story. It ended with the main character saving the Domain Name System but being framed and going to jail for another escapade.

I promoted this book like nothing else in my career. I mailed everybody I knew who had the slightest interest in free software or free culture. I mobilized O'Reilly around the project by holding contests in the Sebastopol and Cambridge offices. For the contest, I made a random-looking pattern of black and white squares and asked everybody to download it. I then created a number of similar-looking black and white patterns and asked some colleagues to help spread them all around the Sebastopol and Cambridge offices. Only one sheet exactly matched the pattern people were downloading, and I gave a prize to the people who first found the right sheet. Each office organized a party around the search.

In my mission to promote Hackerteen, I even set up a visit with Scholastic, the major publisher of children's educational content with many books and magazines under their publishing umbrella (including one author you might have heard of named J.K. Rowling). I boldly went to their New York City office and pitched Hackerteen to a staff person. I showed her my favorite panel in the book: a somewhat climactic scene where the main character solves a difficult test to win placement on a team. The character and the test-giver gaze at a screen that erupts with glorious light, lending a religious aura to the scene.

The editor recognized my passion but explained that the Scholastic publishing schedule was very different from that of a computer book publisher. Scholastic plans some eighteen months in advance of release, and matches their releases to an academic schedule. There turned out to be no point of contact where O'Reilly's publishing model could work with that of Scholastic.

Hackerteen failed to take off. The main reason was that we charged too much. Comic books tend to be very cheap, especially when children are the chief audience. The printed books are flimsy and the color is washed-out. Our book had to be beautiful to be

effective, and after we invested the necessary amount to make that happen, we needed to charge $20 per copy. Furthermore, although our characters were teenaged, our colleagues said that the books appealed to preteens who were looking forward to becoming teenagers. Basically, neither I nor O'Reilly had any understanding of the comic book market, and we couldn't break into it.

Marques invested a lot in promoting the book as well. Among other things, he set up a web site where the key technical elements of the book could be explained in simple language for preteens. I recruited many leading Internet activists to write text, and edited them.

I felt sorry for Marques, because once the book was released, he forged ahead and planned a second volume where the chief hacker's girlfriend becomes a lawyer and gets him out of jail. The motives of the bad guys—who had some sympathetic elements—would be further revealed. This volume would have been even better, because we both really understood our message and the plot elements we wanted. Although I cautioned him that O'Reilly was not succeeding in selling Hackerteen Volume 1 and might not sign another book, he flew me down to São Paulo and then even further to Rio de Janeiro to meet his cartoonist.

The trip was exciting for many reasons. I got to visit again the author who had brought me to Brazil in the first place. Marques also took me to the Ipanema district of Rio, famous for Astrud Gilberto's song, and to the museum of contemporary art in São Paulo. The museum happened at that time to have an exhibit about bossa nova, one of my favorite pop music styles, and I could read the Portuguese descriptive plaques. I thought back to this exhibit years later when I wrote my report about free software in Brazil.

Another pop music encounter was a dance that Marques and Gobi took me to. Gobi's wife tried to teach me one of the dance steps, and someone used their cell phone to videotape my dancing. When I saw it, I recognized myself expertly carrying out the dance steps toward the left when everybody else was moving right, and dancing to the right when everybody else was moving left.

We also went to a book fair. Because the year was 2008, many books were being published in honor of the 40th anniversary of the revolutionary year 1968. It was interesting to see how publishers hooked onto the romance of that period, now so far removed in time. As an experiment, Marques placed a copy of Hackerteen on a shelf of some publisher, and we watched a child pick it up curiously and look through it.

But of course, the purpose of the visit all came to nothing. O'Reilly declined to produce the second volume. Marques expressed no hard feelings, and said he'd try to produce it himself in Brazil as a promotion for 4Linux. I have preserved several

precious copies of the first and only volume, as well as some of the T-shirts O'Reilly printed to mark this ground-breaking attempt to bring our values and knowledge to young people.

My last blockbuster: Beautiful Code (2006-2007)

Computer publishers are clench-jawed Gradgrinds. A book is supposed to teach something practical. It must present an identifiable goal for the editor's routine spreadsheet of costs and earnings to pass approval. But what about a book that's just supposed to show off beautiful things? How could that get published?

My acquisitions were already slumping in 2006 when a computer science professor at the University of Toronto, Greg Wilson, came to O'Reilly with a somewhat half-formed idea he called *Beautiful Code*. Although most people think of programs as functional, the programmers themselves—at least the top programmers—see them much differently. Like mathematicians and physicists, programmers can detect in good programs various intellectually satisfying concepts that they associate with beauty. The notion of beauty itself is beautifully amorphous, so if you ask any accomplished programmer to name a piece of beautiful code, they will come up with something different from other programmers. And this is what Wilson aimed to get: a set of marvelously varied examples of beautiful code, providing wonder on each page turn.

Although the field of computer publishing was teetering, at O'Reilly a curiosity for experimentation and a sense of wonder persisted. *Beautiful Code* was approved.

Wilson's connections proved crucial. Experts whose fame would intimidate me or other O'Reilly personnel readily agreed to Wilson's project. The more than 30 contributions we solicited ranged from a fresh look at sorting—one of the classics of programming—from Jon Bentley of Programming Pearls fame, to a bespoke software project that took text input from Stephen Hawking.

To reel in contributors, Wilson stipulated all royalties to be donated to Amnesty International. Because a book with more than 30 authors could not generate serious income for any single contributor, and because coordinating the distribution of royalties to so many contributors would be a logistical headache, Wilson's decision was a brilliant solution. Many authors said that the donation to Amnesty International provided the main incentive for them to sign up.

True, a few recruits were dissuaded by our support for Amnesty International. One, expressing the narrow prejudice of nationalists everywhere, rejected the project because he claimed that AI was unfair to the government of Israel, and a couple others expressed timidity at being associated with the organization. But the only big problem came when our Beijing office chose to translate the best-seller into Mandarin. The office staff there confided to us that announcing the donation of royalties within the book would not stir up controversy, but that putting that information on the back cover could trigger strong criticism from the government. Amnesty International is hated and feared by the mainland Chinese authorities. On the other hand, although not deaf to these concerns, Wilson averred that leaving the information off the cover would constitute false advertising. Finally our Beijing office found a solution that involved referring to AI by just its initials instead of the whole name.

Even though many of the contributors were seasoned authors, my editing made a huge difference. Even Bentley expressed appreciation at my comments, which included an alteration to his code. Wilson and I handled input files in just about every format known to computing, from TEX to FrameMaker, and shepherded all the authors toward our deadline.

Beautiful Code was such a runaway success that O'Reilly created a whole web site devoted to it, with its own domain name beautifulcode.oreillynet.com, and launched a new series based on the idea of "beauty" in software. I myself contributed a story of achievement and protest at my former employer MASSCOMP to a book called *Beautiful Teams*. But none of the follow-on books sold well. *Beautiful Code* stood apart from all the rest in its fame.

The book took the prize in an important event called Jolt Awards, run by the well-known magazine called Dr. Dobb's. (A few years, another book I edited, this one interviewing famous compiler developers, won another Jolt Award.) The day that I was scheduled to pick up the award, I heard that my mother died after a long bout with cancer at the age of 93. Nevertheless, I showed up for the reward and made a short speech praising the authors as the real winners. I cut my appearance short at the party that followed.

One of my last important contributions to book publishing took place in 2011 with another unique book proposed by Wilson: *Making Software*. Having spent much of his career in academic computer science, Wilson told me he was alarmed by the exaggerated claims often made about the efficacy of various popular or novel programming practices. He was frustrated by the assertions programmers made about the right way to do software engineering, usually with no evidence to back them up. When a new fad—or more often, a variation of some forgotten fad— receives attention, such as Agile programming or SCRUM, or when a new programming language takes hold, or even when a simple practice spreads such as pair programming (where one person watches over the shoulder of another and comments on their coding choices), people line up with testimonials or condemnations that are

completely anecdotal. Entire books promote practices with no rigorous research, representing only the fervent testimony of a programmer who has found success.

Fields such as medicine and engineering have become fundamental to modern life precisely by eschewing such superficial impressions in favor of rigorous testing. Given the importance of software and the scarcity of proficient programmers, software needs some of that rigor.

Our book therefore aimed at a lofty goal suited to Wilson's professional integrity. He gathered leading researchers in software engineering to write chapters for an anthology about what works and what doesn't. Wilson and I collaborated as we had earlier on Beautiful Code, with him recruiting and me doing most of the editing.

Making Software offers a window into this research world in the form of literature reviews by recognized academic experts. The title conveys nothing of the book's modus operandi, but we could think of nothing really apt. The book's poor sales may be explained less by the title than by limits that its honesty placed on its claims. Software is nowhere near being a comprehensive and well-tested doctrine that can dictate practice. Our book could do little more than prove that many of popular assumptions are wrong and that we should not take much for granted. Software engineering research has turned up very few answers to the pressing questions programmers have: how to organize a team, what programming language to use, how to test, etc.

So our message was not appealing. Of course, most research in most fields remains inconclusive; that's why confusion marked the world's response to the COVID-19 pandemic for months. Too often, humanity's plea for answers goes unmet.

Making connections in the public interest (early 2000s)

The most urgent skill needed by a publisher is not editing, but networking. We have to stay in constant touch with the people pushing technology forward. To publishers, "cutting edge" feels as visceral as the actual touch of a blade. The people we cultivated dropped insights into our conversations like nickels into a beggar's cup, and those insights might be in some area quite unrelated to the expertise they gave us for their books or conference presentations.

For instance, I started hearing around 2000 that very geeky developers were buying Apple Macintosh desktop and laptop systems. This intrigued me, because at that time, the Mac was popular among designers and similar artistic professionals, but hard-

core computer professionals passed them by, finding them deficient in the tools the professionals needed. Among my crowd, professionals used GNU/Linux systems. What led to the sudden enthusiasm for Macintoshes?

I'm sure the shift came because Apple based its new system (MacOS X) on a Unix-compatible kernel. They took some oddball version of BSD and altered it further to make a version of Unix they called Darwin. They even released this kernel as free software. Suddenly, the powerful developer tools that had long dominated the field, thanks to Unix workstations and then GNU/Linux, were a quick port to the Macintosh. My company also shifted to Macs, although I didn't get one until I found my GNU/Linux laptops lacking in the functions I needed. At the Open Source Convention, where we still provided a large room full of desktop systems with Ethernet connections for people without laptops, I saw one year that these were all Macs.

In my daily blog posting, I noted the incongruity of setting up proprietary desktop systems at an open source conference. This was nearly the only time I got in trouble for blogging, after doing it weekly for many years. A conference manager noticed the observation (not really a complaint) in my posting and contacted me to explain that the room full of desktop systems were part of Apple's generous sponsorship for the conference, and that we must not anger them. The sentence got stripped from the posting.

Well, tact is not my strong point. I write notable articles because I notice things, and then note them vigorously in my writing. In this case, I admit, I violated not only corporate financial calculations but basic etiquette.

Only one other time, at least five years later, did my penchant for broad observation get me in trouble. Tim O'Reilly and the company had been following reform efforts to insert data processing and digital communications into government, and decided to take advantage of the gathering movement toward transparency and scientific government.

Carl Malamud, a technologist who came very early to the call for opening government data, spoke at our Cambridge office back in the 1990s. Others came later with digital solutions to common problems, such as the FixMyStreet app. This app did nothing more momentous than making it easy for citizens to report potholes or trash, but it offered the chance for local governments to build trust among their residents.

Efforts like these cascaded. I was well poised to join O'Reilly's efforts, having met Beth Noveck in 2005 and followed her work on Peer to Patent into a general fascination with open, digital government.

Tim himself wanted to take a leading role, and built tight relationships with technically sophisticated leaders in the Obama White House (where Noveck also spent a couple years). As he spoke and held his usual rounds among leaders in the field, the company struck up partnerships and sponsorships. One of the new organizations that made a splash was Code for America, which took a leaf from the notebook of Teach for America. Young people sponsored by Code for America would work with local governments to solve problems through the development and dissemination of apps. The founder of Code for America, Jennifer Pahlka, took on a number of leading roles at agencies and married Tim.

The term Government 2.0 came to cover this combination of public engagement, transparency, and better living through data crunching. I believe Tim himself invented the term, although I cannot be sure. Tim had definitely invented the term Web 2.0, which helped observers of business and technology in the 2000 decade understand the burgeoning role of private contributors to the Internet commons.

The 2.0 moniker, which came from the version numbers bandied by computer developers to flaunt important improvements in software or hardware, had been applied metaphorically in other settings to say, "Look up, folks! Things are different now. Out with the old, in with the new." This kind of fad, where catchy terms skip from one context to another, is common. In the 2010s, after some programming teams decided to combine development and operations (deployment) under the catchy term DevOps, other people felt impelled to add the suffix Ops to everything, whether it made sense or not.

After Web 2.0, it would be easy to slap 2.0 on other everyday terms. When our company got interested in health IT, we were non-plussed to learn that Health 2.0 had already been trademarked by another organization. It was a strong and innovative group, led by Matthew Holt and Dr. Indu Subaiya. I attended many of their conferences and wrote for their blog. O'Reilly negotiated to buy their company, but couldn't come to terms both sides liked. And a couple years later, O'Reilly was out of health IT.

Government 2.0 didn't last long for us either. Behind all the bright lights and blogging, our business strategy was to sell our books and other offerings to government agencies. But they weren't buying. Many people would express interest in the themes of Government 2.0 but couldn't find forty bucks in the budget to buy a book about it.

This was truly a major company focus for a couple years. In the thick of our most earnest efforts, I wrote a blog posting where I said that the term Government 2.0 was misapplied. Having just seen a shocking exhibit of frescos from ancient Assyria, I suggested that empire was the first government and that the Greek democratic innovations were the real Government 2.0. What was

my point in churning up all this ancient history? That our efforts at rationalism and transparency were just a continuation of the democratic ideal, and a modern way to bring it closer.

This time I got email quickly from Tim himself. He explained that he and the company had invested a lot of money and branding effort in the term "Government 2.0" and couldn't allow my posting to undermine it. Apparently, Tim had access (or asked our web team for it) to all postings on the O'Reilly blog, because he altered my posting to smooth out the reference to the Assyrians and Greeks. I admit that Tim, having a degree in classics from Harvard, might know more about the history of Greek democracy than I do.

So those were the only two times, during a couple dozen years over which I wrote hundreds of postings for their blog, when O'Reilly managers took out observations because they went against the company's business interests. I don't mind that at all. I realize that hard-hitting and insightful reportage is like sending a shopping cart full of merchandise careening down the aisle of a retail store. If someone has put an item in your way without telling you, it can get hit.

But what about the correctness of my writing, in my own judgment? Do I regret anything I put in my blog postings, journal articles, or conference presentations? Yes, three times. One concerns a misunderstanding of statistical techniques, which I describe elsewhere. Another was an article I hastily wrote about a particular patent dispute without adequate research; I received a lot of criticism and took the article down. The third one concerns a policy recommendation I put in an early article of mine about encryption, which I'll describe here.

⋙

> Digital encryption has scared the security forces of all countries since Whitfield Diffie and Martin Hellman figured out how to do it right in the 1970s (and their system has never been proven ineffective—only particular implementations can be broken). Every few years, some security agency suggests that encryption be banned entirely, or weakened in some complicated way that the agency thinks they can push through because it's too confusing to alarm the public and draw criticism. Sometimes they claim that criminals won't get in between the private sender and recipient, but only the beneficent agencies as "trusted third parties".
>
> The agencies have always failed to persuade the public that we're safe with their hacks to the system, because people who really understand cryptography—people who devote their lives to understanding the algorithms and associated

> risks—are naturally those who want cryptography to work. They swarm over every proposal and demolish the pretense that it can preserve privacy for the average user while giving law enforcement the access it wants.
>
> Back in the 1970s and 1980s, the situation was already contentious, with the U.S. government imposing absurd export restrictions on software that people outside the U.S. could use to protect communications. Did our government think that no one outside the U.S. could figure out how to implement encryption software? The only thing accomplished by those regulations (which eventually were successfully challenged on free speech grounds) was to give headaches to people working across borders on free software. Similarly useless restrictions were later placed on software that could read copyrighted media, such as digital movies that were "protected" with weak key systems.
>
> The debates have become even more urgent with the advent, on the one hand, of digital commerce and online banking where everyone needs strong protection, and on the other hand, of far-flung terrorist networks who have become expert at bringing new recruits onto hidden communication channels.

This is where my mistake comes in. Amidst all this brouhaha, in the mid-1990s, I was asked by some UK-based lawyers to join them in an article criticizing attempts to control encryption. I was quite new to policy issues, and unused to thinking through ramifications of policy proposals. I trusted these lawyers, who had admirable values and impulses. It was also a rare opportunity for me to put forward my name as a co-author in a research journal. So in June 1997, the Journal of Information, Law, and Technology published our article with the righteous title "Can the Trusted Third Parties be Trusted? A Critique of the Recent UK Proposals."

The critique was standard stuff for Internet activists, and remains valid. The problem was that some author had thought up a solution they thought was cool. In their proposal, everybody using a key would have to submit it to a secure archive, where governments could subpoena encryption keys from suspects in the same way they get search warrants for homes and businesses.

Unfortunately, this exact solution emerged later in a U.S. government proposal known as the Clipper Chip. Even though I doubt that those policy-makers had read our article in a minor U.K. journal, I am embarrassed that I went along with this solution. The Clipper Chip proposal was instantly blasted to smithereens by the crypto community. In retrospect, not only was it woefully insecure and intrusive, but I can't imagine the logistics of such a proposal as I generate new encryption keys two or three times a week for various web sites. The worthlessness and absurdity of this "key escrow" idea does not protect it from being regularly raised in fresh settings by new policy-makers.

This excursion into Government 2.0, Health 2.0, and encryption was, I hope, interesting because they covered some crucial historical moments, but I got off topic. Remember what this section of the memoir started off talking about? Before I got into the origins of MacOS X, I was discussing the role of networking.

Among the many authors and fellow editors whom I now count as friends, I formed an unusual relationship based on our shared appreciation for literature with Dinesh G Dutt. This network expert started out at O'Reilly writing two reports on old networking protocols that have emerged with new importance in modern data centers. He then proposed a full book on how to design and configure a data center. I was honored to be assigned to all these projects. Dutt was a central figure in networking; he had actually created one of the protocols he was documenting for us.

After he expressed his dream of becoming a strong writer, we started to discuss literature and poetry. I wrote a poem using the themes and terms I found in his book, and dedicated it to him. He in turn liked the poem so much that he asked for permission to include it in the book, and the managing editor agreed. This became the first O'Reilly book to contain an original poem.

From stories such as this, it will be clear that the friendships I formed with authors were strong and productive. I was repelled by the way some publishers would talk about having a "stable of writers". My authors were human beings to me, and our work embodied many aspects of humanity.

One can well ask where the women are in all these creative encounters with technology. O'Reilly certainly found fewer female authors than male ones, but we published several significant women. One author with whom I worked was Elecia White, who joined a series we were publishing on embedded systems with a very personal approach. Her book *Making Embedded Systems* seemed to me exactly what an entrant to the field would want after getting a degree in electrical engineering or computer science. White bridged the gap between book knowledge and job performance by presenting a grab-bag of important skills such as reading schematics and balancing resource constraints with requirements.

White's creativity was a risky in the marketplace, just as for a novelist whose book that doesn't fit neatly into some genre such as mysteries or historical fiction. Publishers and booksellers don't know how to advertise, shelve, and otherwise promote something that falls outside hidebound categories. White's book didn't stick to the traditional scope of embedded systems. In contrast, most of our embedded series books—which I had proposed and edited—concerned Linux or its derivative Android, and stayed within their technical boundaries.

In addition, White's book came out in 2011 as our embedded series was in decline. Even while home devices and the Internet of Things were rendering embedded systems increasingly a part of our lives, readers were turning away from tried-and-true titles we offered in the embedded space, and often a highly popular book would fail in its second edition.

So sales of her book started out slow, and I was disappointed. However, in her review of this memoir, White tells me that sales have been steady, probably be because her book covered useful, general topics and didn't become obsolete like so many technical books. In the ensuing nine years, she has enjoyed some $30,000 to $35,000 in royalties.

Old friends: Digital Equipment and DCE (early 1990s)

The topic of this section allows me to stand back and sweep my eyes across my career to establish the context in which I came to O'Reilly. When I accepted the job, I thought I was leaving the computer industry proper and taking on a diminished role as an outside observer from the publishing industry. I was unfairly denigrating the company and my own capabilities, because O'Reilly uniquely participated in the communities we wrote about. But now I understand that my career move represented a more significant shift, moving with the computer industry as a whole. Coming to O'Reilly in 1992 allowed me to escape the failed computer development model of the Open Software Foundation (OSF), which I described in another chapter, and to take on a winning model.

OSF itself looked like a way to escape the declining environment for high-performance computers at MASSCOMP. My former boss from that company, Steve Talbott, tried to recruit me for O'Reilly at that time. However, I decided instead to try out Hitachi, one of the world's major corporations that was attempting the kind of hopeless project to which I myself am partial. One of Hitachi's less notable divisions was sputtering along by selling a mainframe computer compatible with IBM's classic line. By joining the Open Software Foundation and adopting its Digital Computing Environment (DCE), the division hoped to drink deep from the fertile Unix watershed. DCE came as a new ripple in the decades-long stream of contortions by the computer field to build powerful networks from systems that had been designed since the 1940s to be stand-alone processors.

OSF was located in Cambridge, Massachusetts, so it seemed reasonable to Hitachi to open a small office in the suburbs of Boston where they would implement DCE. I lasted there for eighteen months; perhaps another year or two was needed to kill the project entirely at Hitachi while OSF itself imploded.

I am happy, though, for the time I spent at Hitachi. The company and its managers were compassionate and intelligent, and my time there yielded many lessons that steadied my path through the future years.

At Hitachi, I had access to a lot of classic Unix source code. I became used to consulting programming code for the products I was documenting. This proved useful at O'Reilly, particularly for the books I edited on the Linux kernel. I soon overcame my earlier intimidation when facing the high priests of the software world, having seen some incredibly ugly and convoluted contributions that were written by the original BSD developers and were now powering computers and the Internet everywhere.

Also at Hitachi, the small writing team quickly saw my facility at learning and developing tools, so I became the go-to person for all kinds of technical problems in building documents using the documentation utilities that OSF provided us. These were, not surprisingly, as buggy as the rest of the software our development team was expected to bring to market.

Hitachi management recognized the importance of training American staff in the culture and mannerisms of Japanese companies. The philosophy and practices I picked up during formal and informal instruction made me more thoughtful and tolerant. I believe my short stay made me more patient with everyone's differences and more able to adapt to a global culture and economy, which the Internet and the Washington consensus (remember that?) were quickly bringing us.

Not long after I started at O'Reilly, DCE came back into my life. This time I did not mind. The work was planned around quality and around meeting users' needs, and hence proved much more rewarding. I was given responsibility for a new series desired by Digital Equipment Corporation, one of the most important New England computer manufacturers, and both a contributor to and seller of DCE.

DEC and DCE—these went together in my life now. I'll refer to DEC as Digital to ensure that abbreviations don't get confusing. I knew the company quite well, as my wife Judy had worked just a few years before at the famous mill where Digital began in the town of Maynard, Massachusetts, and I had many friends within the company who kept me updated on their strategy, offerings, and stumblings. Digital facilities sprawled across Eastern Massachusetts and Southern New Hampshire at the time of their partnership with O'Reilly.

I had long admired the documentation Digital produced for their largest product offering, the VAX/VMS series. The manuals were written not only simply and clearly but with panache. Even more important, the books showed the marks of being written by people who actually used the systems, and loved them.

Now Digital wanted to furnish writers to produce books on DCE for publication by O'Reilly. These writers were happy to accept my editorial guidance. We got along great, feeling like a single cross-company team. Digital was known for matrix management, where people officially cross organizational boundaries (such as between development, testing, documentation, and marketing) to build products collaboratively. O'Reilly's informal modes of collaboration fit in perfectly. Results were high quality.

I drove regularly to New Hampshire to meet my Digital colleagues. I got very friendly with authors Ward Rosenberry and John Shirley. Rosenberry brought his family over to my house and our kids played together. I also basked in the generous aura of Frank Willison, manager of the Digital team that was writing our books.

I believe that our first DCE book included, in its preface, the author's acknowledgment of support from his gay partner. Demonstrating a precaution still appropriate in the mid-1990s, I double-checked to make sure the author understood that this book was going to be sold publicly.

»»

DCE failed because it sought too hard for the Holy Grail of computing, which is to make multiple systems in different places feel like a single system. Some progress toward that goal had been made in the 1980s and 1990s through networked file systems, which included Sun Microsystem's NFS and a competing system called SMB that Microsoft licensed and that proved more functional than NFS in the long run. Another important feature of distributed computing is the distributed register, which also exists in bifurcated form: LDAP in the free software and Unix world, and Active Directory (based on LDAP and somewhat compatible with it) in the Microsoft world. These allow a single source of truth for important information about employees and departments throughout an organization. I edited a book on Active Directory during my sabbatical from Linux and free software in the mid-1990s.

Another monster called CORBA (the creators couldn't even force the title into an easy-to-pronounce acronym) attempted to extend the darling programming model of the 1990s, object-oriented programming, across network boundaries. CORBA reminded me of OSF products, because all the big companies clustered around the initiative but proved incapable of true collaboration. As every company's implementation differed and they never worked together—integration being of course the whole point of CORBA—the project kept adding higher and higher levels of the putative standard, each layer striving to harmonize the incompatibilities of the layer below and, in Sisyphean fashion, creating a hotbed for new incompatibilities.

> And yet the industry hype around CORBA was so huge that O'Reilly set its editors to the task of signing a book on CORBA. Existing books were as indecipherable and gangly as the product itself, and I saw no way to treat the subject in a comprehensible manner. It also moved too fast to tie down in book form—everything seemed to be outdated as soon as it was published. Eventually CORBA shriveled up before we could find an author. The project demonstrated several traits that turned up in later projects and provided great dilemmas to publishers. In particular, the pace of development in computing picked up a great deal in the 1990s, so books were not keeping up and the professional computing public was coming to depend on the richer and richer online information sources.
>
> The residual embers of DCE's impact remained, ironically, in Microsoft's products. I know of just one useful invention that DCE promoted and that remained important to computing: the UUID. This was meant to solve a fundamental requirement of distributed computing: how to distinguish people, computers, and other items with identifiers that could be generated simultaneously on thousands of different computers but be reliably assumed to be unique. Apollo Computer, which was another towering success of 1980s computing for a brief time, invented the UUID. Through DCE, it became seen as a useful feature now implemented in programming libraries everywhere and employed through computing for sundry purposes.

As one can imagine, knowing the train wreck that was DCE, our books did not sell well. But along the way O'Reilly and Digital worked with Microsoft, which had suddenly decided to boost its appeal in the Unix market by supporting DCE. Their goal was not to become a Unix shop, but to make it easy to tie Unix computers to their Windows systems, which were now running a new operating system called Windows NT that they had built from the bottom up to support the familiar (although every-morphing) Windows interfaces. Rosenberry wrote a book on Microsoft's implementation, with me as editor. Microsoft maintained their interest in O'Reilly and commissioned us to produce their Microsoft Press books for online and print distribution many years later, in recognition that our platform and production process were tops in the industry.

The company Digital was going downhill at the time they were working with us on DCE books. Just as I jumped ship at Hitachi, Frank Willison saw the writing on the wall at Digital and took a job as manager of the editorial team at O'Reilly. He played an important role there for years.

☞ *Drawing from the library stacks: free and open source*

Drawing from the library stacks: Free and open source

Contents of this chapter

No service to free software (2010s)

Free and open source software is available for everyone to use, alter, and share. Communities have developed a new ethos for producing this software collaboratively, a model so powerful—the GNU/Linux operating system being a prime success story—that in the early 2000s, proponents had credible reason to believe that free software would become dominant and push out the other major model for software development. In other words, free software would drown proprietary software, which no one except its developers can view or change.

Instead, Software as a Service (SaaS) happened.

The Internet made it possible for worldwide communities to achieve the efficient, highly responsive development model of free software, and to share the fruits of their labors. Ironically, it is also the Internet that makes it possible to run an application on some server in California or Germany and interact with it from your desktop or cell phone. The Internet honed the weapon of SaaS, part of the popular concept of software in the "cloud", which restored the scepter of world domination to proprietary software.

⋙

Historically speaking, free and open source software has been ubiquitous since the beginning of computing. Free software contrasts with proprietary software, which is can be controlled by one person or company through copyright, trademark, and patent laws. Free software recognizes the rights of the software's developers, but the software's legal license allows other people to use it freely, change it, and distribute their changes. I will release the reader from the pain of learning more about licenses in this memoir. Plenty of people have banged their heads on licenses in order to preserve the legal foundations for free and open source software.

If you use an Android phone, you are using free software, although Google and the phone vendors mix in proprietary software and maintain some tight controls on the phone. Android is based on Linux, which has revived the operations of the classic Unix operating system.

What if you use an iPhone, iPad, or Apple MacIntosh computer? You are also using a system with a foundation in free software. All those Apple systems run on another variant of Unix, based on the classic Berkeley Software Distribution (BSD). Apple released their customized version, which they called Darwin, as free software.

So if you're wondering where free software is used, now you know: It's everywhere. It has run the Internet from the beginning, and manages the web experience of millions of people through the popular Firefox browser.

As an exercise in understanding the effects of free versus proprietary software, think back to the incident in the first chapter when an author based a book on a web development kit that readers could download from one company's site. When that company arbitrarily decided that this kit no longer met their marketing needs, and peremptorily took the link down, the book had to be rewritten. If the software had been free by the definition used here (free to share and to change), the source would have been available, so the author or publisher could have made a copy and there would be no risk of having the rug pulled out from under us. Even that tiny effort would probably be unnecessary,

because other people around the world would make copies and collaborate on keeping it up to date. So we could probably have made a link to a public repository that someone would make sure to keep alive.

Waves of activism have promoted even greater use of free software, especially in schools and government agencies. These institutions have various obligation for transparency, promoting participation, and inclusiveness, which call for free software. One of the biggest initiatives in free software (at least in the United States) took place in my own state, Massachusetts, in 2005. The administration undertook a wholesale replacement of its Microsoft Windows systems with free systems. A friend and author I worked with, Sam Hiser, acted as a consultant. He invited me to report on their work, inviting me back to the State House where I had gone to promote use of the Internet in 1992. Proprietary software creates its own culture and demands on workers, making the adoption of free alternatives a multi-layered undertaking. The Massachusetts project was carefully planned and well executed, but the administration (run by a Republican governor) got into a tiff with the legislature (dominated by Democrats) and they killed the transition.

»»

Let's return to the state of computing in 2020. Most people tap their screens and check their messages with barely a thought that they're connecting to servers far away. If they did, they might consider it odd that the picture they're sharing with the friend sitting next to them has to pass from their phone to a server in Iowa or Bangalore. This is the cloud: servers and data storage so hidden from you that you can't even tell in real time how far away they are.

Although the free software movement, following Richard Stallman's lead, disparages the term "cloud" as a vague umbrella term, I think it's very apt for such a secretive and shifting situation. You may interact with one cloud service, such as movies on Netflix, but Netflix runs its servers in turn on another cloud service, Amazon Web Services. Some companies are expert at providing user services, others at maintaining physical infrastructure. I wouldn't be surprised if, someday, one or two companies that do an awesome job at maintaining the physical systems run all the data centers in the world. The Amazon.com retailers of the world would then lease their systems to run their user-facing services.

Conceptually, using the cloud is a form of outsourcing. The cloud vendor says, "Why not run your software on our hardware to simplify deployment?" in the same way that a professional house painter says, "Why not sit back and pay me to reach the high spots?" And a cloud vendor can pile on more and more services to add value, just as a house painter can offer repairs and other construction work.

Here I'll lay out the most significant service currently offered by cloud vendors—artificial intelligence (AI)—and why their entry into that feverish area of innovation may change how innovation itself works.

As I write, software in the cloud is consolidating around a significant trend: Not only do the results of programmers' efforts run on servers owned by the vendor. but the entire activity of developing the software runs on the vendors' systems. Instead of popular free software tools, programmers use proprietary tools created by the vendor. And this potentially narrows the scope for O'Reilly's content, because the company has always based its success on covering universally available software.

Complex computing advances—the types that shatter old assumptions and push brusquely past the barriers encountered by older researchers—take place in open forums. Such was the case with machine learning, which brushed the cobwebs off an old idea called neural networking. Neural networking was a classic model for artificial intelligence that had brought generations of earlier computer scientists to ruin. Those earlier researchers were shipwrecked upon the limitations of their hardware. As with most software advances, massive speed-ups in hardware let machine learning coast to success in the twenty-first century.

Strangely enough, Moore's Law seemed to be reaching its limit a few years before machine learning took off. Conventional chips weren't really up to the quantities of instruction cycles demanded by this repetitive crunching of huge data sets, but clever AI researchers turned to graphical processing units (GPUs). These weren't exactly a dime a dozen, but they were mass-produced to meet the needs of high-speed display processing, particularly for computer games. GPUs had snuck into all computer systems, even those as small as cell phones. It turned out, unexpectedly, that GPUs were great for machine learning because they could run limited sets of transformation on blocks of data streaming in one after another. And like most great computer technologies, GPUs shamelessly stole ideas out of the distant past—in this case, a seemingly obsolete technology called array processors, which were the sole product of a company I worked for in the 1980s.

After the principles of machine learning were found worthy, both hardware and software evolved to serve it better. Hardware expanded through specialized chips to carry out common operations, such as Google's tensor processing unit. In software, the free and open tools generated by early researchers were embraced and extended by Google, Microsoft, Amazon.com, IBM, and other companies.

This is where the cloud came in. Whereas the first successes of machine learning were hammered out in the sweat of researchers who installed new programming libraries and set up massive clusters of computers, later developers found it ever so much easier to let Amazon or Google run the computers and vet all the software.

At first, the cloud companies simply installed the free software tools with convenient interfaces for access. Then the companies started to develop some pretty awesome tools of their own. Sometimes they open sourced these tools, which they could safely do because in time, the convenience and robustness of their cloud platforms would wear down the determination of nearly any business that tried to deploy the tools on its own.

There are other precedents in society for the expropriation of creative producers. Gay and transgender activists (particularly people of color) like to complain that they invent lots of cool styles and fashions, while all the wealth goes to big companies that steal and commercialize the ideas. Thus it was with machine learning researchers and the free software they created. They drove an AI revolution that was ultimately taken over by big cloud vendors.

Let me be fair. The big companies, such as Google and Microsoft, have contributed enormously to research in computer science. They did so by offering numerous enticements to recruit leading researchers. For instance, two engineers from Google named Jeff Dean and Sanjay Ghemawat implemented the company's MapReduce algorithm, perhaps the kick-off of the big data revolution. (I honored this achievement when these engineers revisited it for a chapter in the book *Beautiful Code*). Microsoft set up independent research facilities that published enormous numbers of articles in peer-reviewed journals—an achievement that drives home the importance of independent research communities.

These companies enabled true innovation when they participated as equals in an open research environment. They consciously protected research from the immediate pressures of business. This model goes back at least as far as the formation of Bell Labs a century or more ago.

Many have commented on the consequences of large companies controlling so much of our computing, particularly around privacy and biased analytics. I would like to pose two specific questions related to innovation: Will development on proprietary systems in the cloud be as fertile as that of free software communities and independent researchers releasing free software? Second, will the propriety innovation reflect the interests of the users, considering that it removes much of the choice offered by free

software? (The choice is reduced simply to whether or not to participate in the platform at all.) While you ponder this, I will explain some of the achievements of free software, and how I navigated the fascinating communities that came together around it.

Free documentation (2007—2011)

People have been volunteering the fruits of their writing ever since literacy slipped away from the stranglehold of the high priests. Online, the power of community documentation—a term I invented to describe this volunteer effort—has been recognized at least since The Well in 1980s Berkeley.

The mystery of why volunteers do this work—and even fervently immerse themselves in it—has prompted research over many decades. Many people have tried to measure and explain the contributions of free software developers, Wikipedia authors, and other volunteers. What's new about my small efforts in this area was a focus on contributions of educational material for software. Such contributions could be as small as an answer to a newbie's awkward posting to a mailing list. When it extends to book-length form, even publishers can take interest. By limiting the topic of my inquiry to computer documentation, I could take on such questions as: Do people who ask questions get answers? What aspects of software make it hard to produce educational material? What kinds of projects are most likely to get volunteer contributions of documentation?

Although community documentation attracted my interest quite a while back in the 1990s, I really started trying to grasp the behavior of communities and to try to exert influence on them in the 2000 decade. My role as senior editor at O'Reilly was fading, so I put in some sputtering attempts to create a consulting business around editing community documentation.

My interest in community documentation developed along with my respect for the achievements of free and open source software. My basic thesis was in sync with multiple trends that fascinated technical, business, and political leaders. Throughout the academic and action communities in which I circulated, the concept of "openness" was being extended from software to hardware, academic and pharmaceutical research, business practices, and government. Although many free software advocates had resisted the term "open source" that Tim O'Reilly and our company were promoting, and although I use "free" most of the time myself, the word "open" offers a great fecundity. It has prodded people in every field and industry to look for opportunities for transparency and the inclusion of diverse voices.

Hence the other major trend influencing my ideas for documentation: the "wisdom of crowds" or "crowdsourcing" popularized by journalist James Surowiecki. Research showed that, under the right conditions, a sophisticated distillation of many views would produce more accurate assessments than a poll of so-called expert opinion. Cynics will claim that this thesis has been contradicted by recent political elections and other mob behavior. But those events actually confirm the research, which also showed how the crowd could be misled by premature influence and unhealthy biases.

Another concern powered my ideas for documentation: the emerging crisis in creative content triggered by widespread Internet access and digital formats. The trend is hollowing out media, as everybody has seen by now.

I believe the world is in the opening act of a great shift away from classic works by individual geniuses, toward evolving creative efforts to which many people contribute. Approaching documentation, I applied this hope to volunteer production efforts. During the early years of the 2000 decade, I conducted research into community-generated documentation.

One project tested the boasts one hears about the superb support offered by mailing lists and forums. Project leaders said that any problem people faced using their software could be solved merely by posting to the mailing list and scanning the replies, which senior members would generously serve up. These leaders furthermore claimed that future novices would be able to find all the answers they needed by searching the archives. This spontaneous generation of help has been called "passive documentation".

I suspected that the reality was not so rosy. In my own archive searches for projects (such as Drupal web development software), I would find multiple contradictory answers, each blending correct details with wrong ones. I also found many outdated answers—and it was hard to tell what was outdated.

So I picked a few popular free software projects and followed a number of threads on their mailing lists—28 threads in one study and 14 in another. In each study, I tracked the answers to technical questions. The first study I titled "Do-It-Yourself Documentation? Research Into the Effectiveness of Mailing Lists". The second was more conventionally titled "How to Help Mailing Lists Help Readers (Results of Recent Data Analysis)".

My findings blasted the pretensions of the mailing list operators. Half of the questions I tracked were ultimately answered, and apparently satisfied the questioner (one could rarely be sure, because the original questioner rarely reported success). For a free, volunteer-driven forum, that's actually rather impressive. But it by no means serves everybody. And a quarter of the questions didn't receive even an attempt at an answer.

In another research project, I tried to determine what motivates people to answer questions online and contribute documentation. I recruited the O'Reilly web staff to help me run a survey, and ran some statistical tests to find the most important reasons. I hypothesized that people participated more for selfish reasons, such as to promote their expertise, than for altruistic reasons. But in my results, it seems that altruism barely edged out the self-promotional motivations.

I say only "seems" because results were close and because I discovered much later that I had made a novice error in my statistics. I assumed that answers on a 1-to-5 scale could be compared as ratio data—that is, I thought that someone who assigned a rating of 4 valued some item twice as much as someone who assigned a rating of 2. This is fallacious, because people don't use rating systems that way. I should have compared the answers as ordinal data, which have a much weaker relationship and would have been much less conclusive. This mistake is only the second choice I regret during some three and a half decades of writing articles, the first regret was an endorsement of encryption key archives that I have discussed elsewhere.

Despite this error, I think the article that I based on my study, "Why Do People Write Free Documentation? Results of a Survey", was a useful contribution to the free software movement. It's a shame I didn't save a copy, because the O'Reilly web team wiped it out and it wasn't recorded in any archive.

Seeing an unrealized promise in passive documentation and other volunteer contributions, I sought ways to organize communities and help them meet their needs. I would compare free software documentation to government funding: Nobody wants to contribute to it, but everybody wants it to be there when they need it. I published web postings on my proposals and gave talks at computer conferences.

My ideas for structuring community documentation were as radical as my vision for boosting participation. Manuals would be entirely passé or would form a relatively small core of an enormous distribution ecosystem of education materials spanning mailing list archives, blog postings, and comments on various sites. Projects would not try to centralize documentation, but would thrive on the scattered contributions of individuals writing on their own sites. I imagined spawning a whole new discipline around community documentation, with benefits throughout the software world. Don't accuse me of low expectations.

Huge problems remained in finding documentation and determining its quality. I was confident that I could address these problems too. I suggested a system of comments where people could indicate "documents to read before this one" and "documents to read after this one". I imagined putting all these comments into a standard data schema and creating tools to crawl the comments

and display learning paths to readers. In concept, this anticipated the learning paths that O'Reilly started generating in the late 2010 decade, although implemented very differently. I wasted untold hours on the schema and an API to underpin the comments and permit the automatic generation of learning paths.

My inability to drum up interest among developers and their companies eventually led me to abandon my ideas for formalizing the work done by volunteers. But O'Reilly also noticed that there was unexploited potential in community documentation, and they showed a willingness to capitalize on my research. I spoke about community documentation at two Open Source Conventions and even once helped O'Reilly try to develop it into a business. We got this opportunity when SAP, the huge business service firm, saw how much its users were contributing to documentation about their tools.

While highly proprietary—in fact, one of the early success stories for SaaS—the SAP company had developed an interest in free software and were avidly creating interfaces to their services for popular languages so that their customers and third parties could expand the available tools. Through various interactions with O'Reilly, to which I was not a party, SAP started to express interest in volunteer documentation. They hoped to recruit their enormous user community to create a knowledge base, which they called "social documentation".

This potential project must have held out the promise of big bucks for O'Reilly, because Laura Baldwin took a great interest in it. She was not yet Chief Executive Officer, but still Chief Operations Officer. I think that she was directing policy for the whole company by then. It's significant that, in 2007, she not only set up several days of negotiations at SAP headquarters in the Silicon Valley but attended all of them personally. She also brought in Andrew Odewahn, who was running a lot of our technical innovation and may have already been our Chief Technical Officer, along with one or two other leaders.

Thus, this meeting was a huge commitment of resources. Baldwin rented a van and shuttled the whole O'Reilly crew each day from the hotel to the vinyl and acrylic boxes of SAP's Silicon Valley meeting rooms. This was an opportunity for team building and drawing closer.

My role stemmed from my volunteer work investigating and participating in free software documentation. My vision matched that of SAP: to tap the interests and expertise of the community by championing many small volunteer projects as a supplement to centralized, official documentation. The philosophy was that users within a community best understand their own needs, and can speak to other users more effectively more than a technical writer hired by the company.

I created a presentation for SAP that included an in-depth analysis of documentation for a popular library of the time (jQuery, widely used to simplify the production of spiffy web pages), to illustrate the strengths and weaknesses of online documentation. I also took notes during meetings and created some guidelines for our further cooperation. My term, "community documentation", was adopted for O'Reilly's proposal.

The SAP managers showed enthusiasm during our meetings, but never pursued the project. Perhaps it was just an ideal that wouldn't have borne viable fruit. But the main problem I remember from these meetings was a personal one: I regularly fell asleep.

The problem stemmed from medication. I have suffered from Tourettes Syndrome since the age of five. Recently, I had accepted a round of the medication haloperidol, which brought on overwhelming waves of fatigue. It took me many months to titrate the dose and find a comfortable balance where the drug suppressed most of my Tourettes-related tics without tiring me too much. In the meantime, I found myself napping many times during the day. I would simply lie on the floor of my office and drift in a dreamy mist for 20 or 30 minutes. People may have seen me on the floor occasionally but did not bother me.

In meetings with a client, of course, falling asleep was highly irregular. Baldwin would send instant messages or email saying "Wake up!" Why I didn't confess my medication issue to her, I don't know. Doing so would not solve the problem, but might have earned me some sympathy. Anyway, I never talked about it. I'm sure my behavior disqualified me from any future meetings with clients. But Baldwin didn't judge me entirely by this shameful lapse. She continued to seem fond of me, treating me professionally and boosting me occasionally with praise.

A couple software projects did allow me to volunteer in my spare time to organize documentation efforts, and some documents were actually produced. Usually, though, these projects decided to go down a more traditional path and just hire a technical writer. It's no surprise that when I laid my vision out before SAP under the aegis of O'Reilly, they demurred.

The volunteer effort in software documentation that showed the most promise was not my invention, but an unlikely venture founded by a New Zealander named Adam Hyde. Hyde was an artist, a unique and inspiring individual who migrated to Europe, created interesting art installations, and learned a good deal of technology in order to spin up modern artistic experiences. Like me, he noted the paucity of good educational materials for the tools he was using, and launched an organization he named FLOSS Manuals. The acronym FLOSS is commonly used for free, libre, and open source software, so he slapped the label on his plucky venture.

The FLOSS Manuals notion of documentation was stodgy compared to mine. They focused on producing a small but useful manual for each tool. The manual would not be fluid and constantly evolving, as I saw software documentation. The book would be created at one fell swoop and then updated through a follow-up project sometime in the future.

I forget how I got involved with Hyde—I believe he approached O'Reilly as a company and that I was the only person to step up and take interest—but I participated in several projects and watched his vision gel into a rigorous documentation process he called a "book sprint". Sprints were already common on software projects to accomplish focused tasks through group participation. After Hyde experimented with approaches of varying sophistication to book production, he arrived at a strict five-day sequence that FLOSS Manuals applied over the years.

I had no idea where my involvement with FLOSS Manuals would take me—certainly not expecting to cross continents. One of my first projects produced a highly praised book on the command-line interface for the Free Software Foundation. This achievement drew me closer to Richard Stallman and others in that institution. I should mention here that I've contributed to other, more traditionally generated manuals published by FSF, and consider many of them to have impressively high quality. My name was even on the cover of the GNC C Library manual for several years, and I eventually asked the organization to remove my name because I had written up just a few APIs.

I traveled to California and met with some 20 other FLOSS Manual volunteers at the GooglePlex for a Winter of Documentation that Google sponsored in conjunction with their well-known Summers of Code. (The so-called Winter actually took place in the Autumn, which I remember because I enjoyed seeing the sukkot that Jewish employees set up on the Google campus.)

Google was generous. They took care of us for the week, paying for expenses and providing us space, staff support, and meals. They imposed no conditions on FLOSS Manuals' work. I could imagine Google objecting to the use of their facilities to work on documentation for OpenStreetMap, which might conceivably emerge as a competitor to Google's own map service, but there was no such intrusion.

One detail in Google's planning had an outsized impact on the volunteers: an incomprehensible lack of lunch diversity. Every day, the same sandwiches turned up. We all grumbled about it to each other, and I got a lucky reprieve. One of my authors, Steve Souders, was currently working at Google (a natural next step after the company where I first met him, Yahoo!). He came for me one

day and treated me to lunch at the famous Googleplex cafeteria, where different stations served up foods from many parts of the world. You see the same plenitude in upscale college cafeterias nowadays, but I think Google was an early example.

At the Winter of Documentation, Hyde assigned me to work on a book about the free software KDE desktop. In this context, a "desktop" is a software package that helps programmers produce consistent and highly capable graphical interfaces. I evaluated the contributions that my team members chose to make and declared that this book wasn't a technical manual but a guide to joining the KDE development community, this focus was accepted by the team. Most of them were very talented young developers from India, one was still a student. We not only finished a nice document but bonded during that week. We parted with sadness and mutual appreciation, after I drove them to San Franciso in my rental car.

I chronicled all this work with copious articles written on the spot. I described book sprints and contrasted the FLOSS Manuals approach with conventional publishing, which differed in almost every detail despite the goals they had in common. Most of these articles were happily published by the O'Reilly staff on our web site, although I think they have all vanished now in one of the web team's blind purges.

In our most extensive endeavor, half a dozen FLOSS Manuals volunteers traveled to Amsterdam in March 2009. We had been invited by The Institute of Network Cultures to a conference called Winter Camp, housed at a college on the Eastern edge of the city, to explore with new forms of organization. Although I never really learned what the Institute of Network Cultures, did, I got an intriguing education in cultures at the conference. We were asked to hold meetings where we could distill our methods and share insights.

One really cannot grasp this unique event without being part of it: everything from live presentations to the evening meal we cobbled together after suffering through terrible cafeteria food for several days. (The meal we made was actually no better.) One might be able to derive some of the feel of the conference by watching the video interviews conducted there by Gabriella (Biella) Coleman, an anthropologist who achieved fame by analyzing voluntary online communities such as Anonymous and the Debian GNU/Linux project. Do not watch Coleman's video of me, however, because I came in fatigued by jet lag, sleep deprivation, and excitement, so my Tourettes syndrome flared up and my attempts at articulate presentation were disrupted by sequences of nervous tics.

In addition to Coleman's videos, the conference led to a book: *From Weak Ties to Organized Networks: Ideas, Reports Critiques*. It includes three sizeable blog postings I wrote for the O'Reilly web site from the conference.

Our campus was in an outlying district of Amsterdam where the city had relinquished traffic lights or any other form of vehicle control, save for a slightly raised crosswalk that provided a gesture of recognition to pedestrians. We occupied bunks in rooms designed like youth hostels, slept with five or six bunks in the room—meaning we did not sleep—and washed in a shower provisioned with one infinitesimal sliver of soap. We shared the cafeteria with high school students on some camp experience, because the conference had deliberately been scheduled when university students were on vacation and accommodations were at their cheapest.

Winter Camp, roughly put, explored ways of exploiting the Internet to conduct social change and build diverse, inclusive communities. The participants were all politically left of left, and many groups boasted of being "autonomous". I don't know how autonomous they could be, given that they took the same buses and ate the same atrocious cafeteria food as the rest of us. Most attendees enjoyed some academic position that permitted them to explore their autonomy freely.

While the conference participants staunchly opposed oppressive systems around us, we all benefited from our privileged place in the world. Our educations and skills allowed us to take our chosen directions in life, and to live wherever we wanted. I noticed during my conversations with other participants that many had been born on one continent, received advanced degrees on another continent, and were currently working in a third. It might have been hypocritical for them to critique the hegemony of the global elite—but I do appreciate that they were in a position to understand the manifold problems of our world, which travel even faster than they did.

Lost history: Why did the iPhone store open? (2007—2010)

This chapter covers various achievements by free and open source software communities. I'll turn here to a change in the way billions of people interact with computers, and why they can thank free software hackers for it.

One of the most world-shaking computing advances of all time was the development of the platform for mobile devices, where outsiders could offer their own applications to owners of the devices. These applications (so common they are called "apps") have transformed the way almost everyone leads their lives. Because the device's owner can get apps from other places besides the

device's vendor, the sources for apps are often called "third parties". Restaurants and stores offer apps to let visitors place orders before picking them up—a particular relief to have during COVID-19. A conference can create its own app to help you manage your schedule, and a museum can create a virtual docent to accompany you through its holdings. Third-party platforms are an invitation to the whole world to innovate.

It would be hard to exaggerate the value of the platform in bringing digital benefits to the world. Innumerable companies and web sites have adopted the idea of a platform for developers, unasked, to offer applications. A platform means that people can use their familiar setting—whether it be a mobile device, a social media site, or some other digital place—to run the program provided by an organization such as a restaurant or museum. Most platforms have restrictions, whether to protect visitors from malicious programs or to promote the interests of the platform owners, but they still represent a great advance in human communications.

But few people know how these platforms came into being. This is a story I've often told, and it goes back before what most people think is the beginning of the story.

Most people remember that Apple's iPhone was the first significant technology to offer a third-party platform (subject to oversight by Apple). For years, whenever an organization wanted to bring in others to contribute to its community, it would say, "We want to build the iPhone store for..." whatever community they were representing. The iPhone store thus became a catch-all phrase for openness and crowdsourcing.

But why did Apple create this store? When the iPhone was released in 2007, the company announced that they would provide all the apps themselves. They couldn't countenance offering space on their screens for other creative developers.

Some six months later, they made an about-face. Steve Jobs announced the iPhone store. Most observers didn't know why. The trade press and high-paid consultants came together around a feeble justification: Jobs and his company must have decided their go-it-alone strategy was limiting and that bringing in new energy from outside would enhance the product.

But the truth is fascinating for what it says about free software and computing communities.

When the iPhone came to market, hackers around the world quickly grasped the power of a general-purpose computing device that could fit in your hand, especially when it could connect to the Internet through cell phone towers that were nearly

universally available. These tinkerers wanted to realize their dream applications on that device, in the same way that the 1970s idealists in Steven Levy's book *Hackers* glommed onto any digital system they managed to walk by.

Apple tried to maintain tight control over the iPhone, but the company had not reckoned with the power of free software. Some basic design choices left the door open.

»»

As in recent versions of the Macintosh computer, Apple used free software to speed up development of the iPhone. Why reinvent things such as multiprocessing and memory management when these basic capabilities had been solved in free software decades before? Having chosen a version of Unix as their operating system, Apple hid within the iPhone all kinds of convenient tools used by Unix hackers over many decades. The free community quickly uncovered these tools and used them to explore the iPhone down to its guts. Along the way, they figured out how to break the security that Apple hoped would prevent them from loading their own apps.

But how could they develop apps, not knowing the functions provided by the iPhone? Here again Apple laid out a red carpet welcoming the hackers, through another design choice. Jobs had settled on Objective-C as his preferred computer language many years before, when he started NeXT Computer. He seemed to harbor a fondness for this language that basically no one else was using, because he came back to Apple and offered it as the main programming language on the Macintosh as well as the iPhone.

Objective-C may be obscure, but its design makes information hiding harder than in some other languages. Because the language does a lot of run-time evaluation, determined programmers can find function names and arguments in plain text within binary files. Objective-C is also supported by the popular free GNU compiler from the Free Software Foundation.

Result: Within weeks of the release of the iPhone, online communities were exchanging programs and running them on the devices.

Apple knew quite well what was going on. Whether the trade press and consultants knew, I can't tell. But I know that a tremendously talented hacker and (ironically enough) security expert, Jonathan Zdziarski, wrote a book for us documenting the jail-broken iPhone interfaces. Another author wrote a similar book for another publisher. I also penned an article about this amazing community effort for the O'Reilly web site in January 2008. I did so partly to promote our book and partly to make sure the history

was documented (although like most of my historically significant articles, this one disappeared from the O'Reilly web site during a reorganization). You can now find my original article, retrieved through the Internet Archive, on my own web site.

So Apple had no choice. The richness of unauthorized apps coming out from the community would eventually win over the public, contrasted with the paltry offerings from Apple. So they created a new, public API, and invited the world to contribute. The age of software platforms took off.

There's an even bigger story I wanted to tell, but didn't get the chance. I was angling to get this story into Steven Levy's classic book *Hackers*.

Now you're thinking, how could I have worked on that book? Wasn't it released by Doubleday Publishing in 1984, a full eight years before I got into the publishing field? Indeed, when *Hackers* was published, the only role I played was that of an enthusiastic reader. But when Levy decided to re-release the book in 2010, O'Reilly managed to land the contract.

Neither Levy nor O'Reilly management wanted to actually update the text. I urged Levy to add a fourth part to reflect events I found crucial. Levy had finished his 1984 edition with an epilogue that came to feel off-key over time. He focused on Richard Stallman, but as I read it, presented him as a lone, sad figure wallowing in his MIT office. Even though Levy took another look at programming ten years later, he didn't seem to understand the incredible tidal wave of free software in our time, spearheaded by Stallman, and I suggested he give it the coverage worthy of a book called *Hackers*. But Levy wasn't interested.

The one thing we could do to enhance the new edition at O'Reilly was to add links, the way journalists offer doorways from their articles onto the Web by attaching links to key phrases. Mike Hendrickson and I spent hours doing this for *Hackers*.

It was great fun. Levy referred to events of past centuries that I suspected were unknown to many readers, so I put in links to history pages. I also tried to find a link for every company, university, or other organization mentioned. (I did not point to Wikipedia, because it isn't a primary resource. I also refuse to include links to Wikipedia in works that I write or edit, although I allow authors to refer to phrases from Wikipedia as evidence of common conceptions.)

For *Hackers*, our most diligent sleuthing took place on my least favorite part of the book, the part that discusses early video games and their creation by companies presented in the book as exploitative and dissolute. Hendrickson and I found that most of

these obscure video game machines, once widespread commodities, continue to be playable through software simulators available on the Web. Who developed all those simulators? Talk about dedicated hackers!

Colleagues and community (2000s)

I suppose people working in any field can form vibrant communities, but the free and open source software movement is especially conducive. Participation encourages personal and communal trust that spans decades and continents. This springs partly from idealism, but we must beware of attributing everything in free software to idealism—such an attitude downplays the movement's resilience. A bigger contributor to the mood of free software is its collaborative ethos. The process of contributing is like an apprenticeship in how to care for others in one's community.

I'll pause here while listening to the shouts of scoffers who point to chauvinism of various types, nasty arguments over trivia, and other anti-communal behavior in free software communities. Yes, we acknowledge when bad things happen. But please also acknowledge that nastiness can appear widespread when it is actually confined to a few participants (as on the cyber-rights list I started for Computer Professionals for Social Responsibility), that many well-meaning people are trying to control destructive behavior, that destructive behavior usually ends by destroying communities that can't control it, and—this may be hardest to accept—that rancorous words may emerge from a deep reserve of love.

Healthy online communities have taken time to develop. But members have gradually found ways for the community's majority to exert its will, snuffing out the abusive and manipulative assaults on character that observers deplore on today's Internet, and that actually have troubled online media from its early years. Free software communities have developed some of the most sophisticated strategies.

Because I was not a member of any of the commonly excluded groups, my own experience consists of warm associations with authors, tech reviewers, advisors, and leaders of the projects I worked with.

At one conference, while people threw their tired limbs across couches in the lobbies during one of the long evenings that followed the day's formal sessions, I held a conversation with a free software user and consultant named Zak Greant. I'd put him on several projects as tech reviewer and saw him as very thoughtful and congenial. Greant started to confide to me about his his bouts of ADHD, depression, and burnout. This was years before people generally understood how widespread and normal such ailments are.

But I had worked for a few years in the mental health field and approached depression without stigma. I listened closely to Greant and gave him support. In his review of this memoir, he wrote, "I should mention how glad I was for that chat nearly 20 years ago. I was really, really struggling. Talking with you made me confident enough to talk with others and that's led me towards much better mental health."

Several authors of mine, also, suffered from depression. I could be open to them as they confided the problem, and guide them toward finding an appropriate role while lining up a co-author.

One author named Arjen Lentz had perplexed me because he had started with great enthusiasm on a joint project with several other authors, and called in regularly to our weekly discussions, but was failing to meet his deadlines. Finally he asked to speak to me individually and confessed to me that he was clinically depressed.

I validated the pain this was causing him and went through a frank discussion of how he could continue to help the project without shouldering more of the work than he could handle. At the end of our conversation he said, "I knew I could talk to you about depression, because I talked to Zak and he said you had been very helpful when he told you about his depression." I had no idea that Greant and Lentz knew each other at all. One lived in Canada, the other in Australia. But the free software movement knows no boundaries, and strong relationships erase all ethnic and national distinctions. My casual good deed—in this case, my willingness to listen sympathetically to Greant—repaid itself years later.

Lentz decided to go public with his problems and create an international support group called BlueHackers for computer professionals suffering from depression. He created small, blue square stickers for this group, and urged people to put them on their laptops to signal their support. At conferences where he spoke, he would ask the audience to indicate who had depression themselves or in their families. Many people would raise their hands, and he would hand out his stickers. I took a bunch and did this while speaking at some conferences too.

Religion often forms a point of contact with other people. I've shared stories of religious practice with a member of the Church of the Latter Day Saints at a party that his company hosted at the Open Source Convention. And I had many such conversations with my manager, Frank Willison. I never accepted the facile admonition to avoid discussing religion with people one knows casually. How could it hurt a relationship to let someone discuss their values, their commitments, and the most significant community in their lives?

Well, maybe it could hurt. At the Open Source Convention, I once asked someone whether she had plans for the summer, and she said, "I'm going for a walk." An incomplete thought seemed to lie behind that statement, so I asked where the "walk" would be and she answered, "Northern Spain." I immediately recognized the Camino de Santiago pilgrimage, and she confirmed she was engaging in that favorite activity for centuries of faithful Christians. She then let me know that she is reluctant to talk about religion among computer people, because they often become uncomfortable or hostile. No community achieves its ideals all the time.

Pragmatism and propaganda (1990s)

It's not my intention to give an overview of free and open source software, but this memoir will make more sense if readers can discard some impediments to understanding the movement. We have to dispose of two myths in particular: that the free software and open source software movements are acrimoniously opposed, and that the Free Software Foundation (FSF) is doctrinaire.

The term "open source" was invented not because any proponents disagreed with the goal of freeing software, but because they thought "open source" was easier to understand and had less baggage to scare adopters. Proponents who stuck with "free software" criticized the new term on several grounds, but that doesn't by any means indicate that the two groups refused to work together. The misconception that they were at loggerheads seriously mars Christopher Tozzi's book *For Fun and Profit: A History of the Free and Open Source Software Revolution*, which otherwise offers fine historical insights.

If the free software and open source movements couldn't work together, Molly de Blanc—a long-time manager at the Free Software Foundation—would not have served on the board of directors of the Open Source Initiative. And how could Allison Randal—a leader in the Perl community who worked with O'Reilly on conferences and other projects—serve as president of the OSI, after being invited by the Free Software Foundation to participate in a conference about the GPL in January 2006, and then serving on one of the drafting committees for version 3 of the GPL? But the myth appeals to people who like simple, black-and-white controversies.

This myth reminds me of facile stories about the 1960s civil rights movement that put the Reverend Martin Luther King, Jr. into contention with nationalist leaders such as Malcolm X and Stokely Carmichael/Kwame Ture. In fact, although disagreeing strongly, all these leaders had respect for each other, consulted with each other from time to time, and recognized that each had a role to play in freeing their people. Similar regard exists between the leaders of free software and open source.

»»

I, too, was at that 2006 conference about the GNU General Public License, or GNU GPL, colloquially known as "copyleft". This GPL was developed by Richard Stallman, one of his acts of genius, and it formed the lynchpin around which everything else in free software revolved. The GPL was adopted by Linus Torvalds for the Linux kernel, as well as many other software projects. The GNU GPL is also the template for Creative Commons licenses in literature, art, video, and other cultural contributions.

Over time, after GNU GPL proponents scrutinized the uses of free software by companies whose ethics were less commendable than the movement would like to promote, the GPL was declared in need of an update. The weaknesses of the license in use by the Linux kernel and others (version 2) were subtle, and the proposed changes were scattered. A huge outreach effort brought in all manner of interested observers to help design version 3.

I thought that the web site set up by the FSF for comments on the new draft license was the best design I had ever seen, and probably have ever seen, for displaying edits by a massive variety of people. One could easily see the exact word or passage each person was critiquing, even if half a dozen people offered comments on overlapping phrases.

The new license was finalized in 2007. Oddly, a lot of people rejected it. The Linux kernel developers decided to stick with version 2. And perhaps that was just as well, because there were hundreds and maybe thousands of developers who owned different updates made to the kernel. Getting them all to upgrade to version 3 would have been logistically tortuous.

But on to the other myth I wanted to dispel. Many people identified the FSF with Stallman, and considered Stallman some kind of fanatic. Stallman has personal oddities, and I believe he stayed on as president of the FSF too long, but it is unfair to call consistent and insistent principles "fanaticism".

Think of how the ACLU or the Electronic Privacy Information Center hammer on any government proposal that would weaken privacy. Some of those proposals address real problems—notably terrorism, violent crime, and pandemics—but it's up to someone to argue at every juncture for our fast-eroding rights, and that's what the ACLU and EPIC do.

Or think of the early Jewish arrivals in nineteenth-century Palestine, bringing with them little but a conviction that they needed to form a Jewish state. One can certainly criticize some of their behavior and tactics, but one has to admit they created an

unprecedented new reality that, before its success, most people thought both unnecessary and ridiculous, rescuing hundreds of thousands of Holocaust victims that the rest of the world was content to let die.

Although many people have heard of Stallman's alleged fanaticism (that is, an insistence on the difficult but necessary steps to preserve privacy and freedom), I think that very few people understand his pragmaticism. I've encountered it a few times while working with Stallman on various policy issues. The first time, he was still living in his office at MIT, where I came in to strategize about how to work on a copyright issue where the Boston chapter of Computer Professionals for Social Responsibility was trying to get Representative Barney Frank's support.

At the GPL 3 conference, members of the audience argued with Stallman, mistaking his pragmatism for inconsistency. (So you see, you can't win. If you're not dismissed for being a fanatic, you're attacked for your inconsistencies.) I'll illustrate his pragmatism through a conversation I overheard at a FISL conference in Brazil we both attended.

The FISL organizers tried to make logistics easy for attendees, housing us as much as they could in a single hotel and providing a bus to take us between the hotel and the conference venue. Unfortunately, this bus service was unreliable. One morning, I found myself waiting for an errant bus with Stallman and two American attendees who worked for the KDE project, a free desktop for Linux and Unix systems. We decided to share a cab, which for reasons I can't remember I ended up paying for. I sat in the front, while Stallman shared the back with the KDE developers.

The developers complained to Stallman that many companies were building non-free applications on top of KDE, and were going on the KDE mailing lists to request advice. The KDE developers found that offensive and wanted to tell all the KDE supporters to refuse advice to anyone developing a proprietary app.

Whether that policy would be feasible didn't come up. But Stallman advised against cutting off the proprietary companies. He said that having more and more applications run on KDE, including non-free ones, would make KDE more useful and hence promote their mission of providing a free desktop. So they should be generous and answer the companies' questions. I consider this a victory of pragmatism over ideology, and one of several examples to refute the claim that Stallman is ideological. (In his review of this chapter, Stallman pointed out that he still wouldn't advise the use of proprietary software—he just saw an advantage to helping the developers master the free platform.)

The FSF's work over the years has increasingly taken on privacy as well as freedom. Their call to computer and Internet users to refrain totally from proprietary hardware and software is too tough for me to follow, certainly. I make heavy use of the cost-free services that Google and other sites offer in exchange for snooping on me. But now everyone recognizes the intrusions that corporations have made on us through digital surveillance and manipulation, vindicating the FSF.

I am lucky to live in the same metropolitan area as the FSF office. In fact, it is located only a few blocks from the Downtown Crossing office where O'Reilly was located during the final years of my work there. I have gone down to the FSF a couple times a year to stuff envelopes or to get volunteer training for their LibrePlanet conference. And I have attended LibrePlanet each year, sometimes as a speaker.

Toward the end of my tenure at O'Reilly in 2020, I came in to the FSF for volunteer work and found the office closed. I met Stallman in the lobby; he too had an errand in the office and was also surprised that no one was in. A third person came, hoping like me to volunteer. Eventually we reached one of the staff by (non-free) phone and were told that they unexpectedly decided to all go out to lunch together. So Stallman asked us if we'd like to have our own little lunch with him, and we jumped at the chance.

Accepting an invitation to lunch was not a casual decision; it required clear-headed principles. Just a few months earlier, Stallman had been caught up in one of those relentless recriminations that take over during highly publicized controversies. He had made a comment about the victims of billionaire Jeffrey Epstein, which unlike his decades of sage advice about freedom and privacy had popped up on the screen of some zealot for the cause of gender justice. We have already seen that Stallman expresses very nuanced views and is often misunderstood even by supporters—so this issue seemed guaranteed to turn into a noxious sore. Complaints by people who showed no understanding of what he said were picked up by the press and spread further. One would hope that journalists would go back to Stallman's original words and try to explain the truth, but that didn't happen. Ultimately, to save the Free Software Foundation from a backlash, he resigned as head of the organization he had founded and inspired.

To be honest, I had seen over the years that Stallman entered into controversies without always understanding the feelings he stirred up. In his review of this memoir, he told me the background behind some of the controversies and persuasively explained the importance of his stance. Still, I expected that eventually he would back away from some of his public roles. What disgusted me was to see it happen over a debased rumor.

Luckily, he was still active in many organizations and remained head of the GNU project, which had pushed free software to prominence in the 1980s and 1990s before the mainstream press noticed it. In fact, Stallman had come to the FSF office today to pick up things he needed for a business trip to Europe. I joked that maybe he could hitch a ride on the carbon-neutral boat that climate activist Greta Thunberg was taking at the time. He responded quite seriously that because he has no children, he creates a tiny carbon footprint in comparison to anyone who has them. Incidentally, another customer in the restaurant introduced himself and asked to have his photo taken with Stallman, so clearly there are people out in the world who still recall his contributions.

One of Stallman's visionary acts was his 1997 story "The Right to Read", which in addition to being quite a well-crafted work of fiction, lays out a chilling scenario that was completely fantastical at the time but has since become an everyday reality. In this story, Stallman imagined critical educational content locked up on encrypted servers so that students needed to go into great debt just to get their educations. Of course, this is precisely where the educational field is now. Stallman was truly prophetic.

Even though I work for a publisher, I'm disgusted with the business of textbooks today. They were a trivial part of my educational expenses back in the 1970s, but now constitute a heavy investment for students every semester. I accept that many of these texts are worth the hundreds of dollars they cost. They are painstakingly written by experts, carefully edited, and enhanced with lavish production values. Many come with web sites whose content is probably valuable. Yet they are financially burdensome.

The right way to create textbooks is this: Educational institutions around the world should band together and set aside funds to employ authors and publishers in the creation of content that is offered for free. The authors and publishers would still be amply compensated, but no one would be denied an education for financial reasons.

But simply in laying out this proposal, I can see why it would not be adopted. The educational institutions have woven access to textbooks into their own business models. If all the content was openly available to the public, they would have to prove that their professors added value and would have to compete on the quality of the classroom experience. And I know lots of great professors, but the pressure might be hard for some.

So where do the charges of fanaticism against free software advocates come from? Partly from the strictures of language, which can be the most liberating or the most controlling aspect of human culture.

Like all marginalized and historically downtrodden sectors, free software proponents need to fight norms so entrenched that they usually go unrecognized. Take racism, which is so normative that it requires constant tagging and critique by people of color and

their allies. And “people of color” itself is an unfortunate compromise with racist reality, because without racism a person from Cambodia would have no reason to feel a special bond with a person from Nigeria.

Language is a central part of norms. It is almost impossible to talk about policy in areas of computing and networking without encountering terms that reinforce oppressive norms, such as “piracy”.

I launched a flank attack in 2004 against the use of the term “piracy” to describe the unauthorized distribution of copyrighted content. I discovered that, for all their wanton violence, pirates in the Golden Age of exploration represented a revolt against oppressive seafaring conditions and conducted themselves with a breadth of democracy that was unusual in their time. My article was thrown away along with most of the blog postings of that period when the O’Reilly web staff revamped their site, but many years later I managed to retrieve the text because, on a lucky impulse, I happened to post the full article to Dave Farber’s “interesting people” mailing list, and it was archived there.

The vast majority of software that technically qualifies as “free” also qualified as “open source”, and vice versa. More specifically, virtually all the licenses approved by the Free Software Foundation are approved by the Open Source Initiative, and vice versa. The terms carry different values whose philosophical, business, and semantic issues have been exhaustively explained elsewhere. So I’ll just mention my own approach to using the terms.

I was happy to adopt the term “open source” after Tim O’Reilly held a historic summit suggesting it as an alternative to “free software”. One simple reason for the change is that the term “free” in English is ambiguous, covering concepts expressed separately by words such as “libre” and “gratis” in Romance languages. Many free software advocates use the term “libre” in English, but that in turn requires explanations to listeners.

I developed a new depth of appreciation for the term “free software” after hearing a lecture by a researcher from the American Association for the Advancement of Science. He told us of his forensic work uncovering massacre sites from the wars following the split-up of the former Yugoslavia. This researcher said that the use of free software to process and present his results was critical, because only if the source code was a public asset would listeners accept his dangerously controversial and provocative findings. I decided that freedom was truly important and should be emphasized in the celebration of this software.

So now I prefer the term “free”, but still use “open source” where “free” might confuse or bias readers. In most articles, I start with a reference to “free and open source software” and then choose one or the other for subsequent references.

In short, free software developers take words seriously. The Free Software Foundation maintains a long web page detailing how to discuss software, licensing, and other aspects of the computer field. Outsiders may think of this obsession with correct wording as the ravings of an isolated minority. But isolated minorities have been persecuted throughout history through the manipulation of language, so they must combat the everyday use of words in order to combat the norms that oppress them. Recently, for instance, transgender people have labored hard to unpack the languages that confine them. The same problem goes for free software advocates in a legal and social terrain dominated by powerful institutions whose intentions toward their movement can be hostile.

Open wallets for open source software (early 1990s)

O'Reilly's whole-hearted adoption of free software in the 1990s centered us in the social impacts of computing. But it also heralded commercial success, because we were often the figurehead leading each ship of the open source Armada into the uncharted future.

In the 1990s, we crept forward on several fronts, not initially seeing the common thread that ran through our efforts. We ran highly successful Open Source conventions and created new series for Linux and the MySQL database. I attended each Open Source convention, blogging incessantly, and edited almost all the books in the Linux and MySQL series. The marketing person assigned to these books, Betsy Waliszewski, communicated with me daily and recognized along with me that there was a strategic opportunity.

Together, we created a marketing strategy for O'Reilly in free and open source software. Conventionally there's nothing new about marketing strategies—but ours was unique in several ways.

First, the field of free and open source software is immense, diverse, and globally distributed. It's one thing to create a strategy for a strictly delimited domain such as the Oracle database or even a field such as security. It's an entirely different endeavor to figure out how open source will affect markets and the next several years of technological progress. We had to be in strong sympathy with the leaders and creators in the various open source fields.

And man, were there free software projects! Tracking all of them was beyond the capabilities of any single person, but we talked to a lot of community members with their fingers on new developments.

We were actually lucky at that time, because the number of projects in free software have multiplied even more over the years. A few facile explanations tend to be offered: Free software is cool, companies recognize the benefits of releasing software under free licenses, developers would rather share software freely than deal with the headaches of commercialization, releasing a package as free allows it to tie in with a powerful ecosystem of other free software packages, etc. Going beyond these observations, I think there has been an explosion of new software in general because new high-level languages make development much faster, and robust test strategies bring high quality within reach of modestly funded teams. These benefits, in turn, are driven by the decreasing cost of high-speed computer hardware.

A second unusual aspect of the strategizing that Waliszewski and I dived into was our commitment to supporting a social movement, not just our own company. We knew that promoting free software to the larger society, and supporting the attempts of free software communities to promulgate their technical information, would benefit O'Reilly as well as the movement itself and society. Yes, we were idealists, and we were right. Other businesses in the free software space maintained a similar balance.

We had an important role mission at O'Reilly because it was hard for free software communities to articulate the implications of their work for the larger society. Some free and open source developers, along with media-savvy supporters, were advocating for the movement. But the business community was not helping much. Companies that built and promoted free software were small and relatively unknown. (Even Red Hat was once small.) At O'Reilly, where communication was the company's fundamental mission, we played an outsized role in promoting free software. Within a few years, major companies such as IBM and Oracle—eventually even Microsoft, which originally embodied everything free software communities hated—would adopt open source and trumpet their support for it. Academics and governments would also come along and discuss the meaning of freedom and openness in software. Waliszewski and I, with support from the broader O'Reilly company, leapt into the arena and carried the torch before it was widely noticed.

This was an electrifying time for me. Waliszewski and I kept our feet down on the pedals of a long-distance race to make sense of free software and respond to the community's documentation needs. Our views aligned. Our books sold well, but we also received the accolades of those who needed our support through conferences, documentation, and my recurring blog postings.

Many companies chose to present new products and services at the Open Source convention—particularly after the closing of the major Linux convention, the Linuxworld conference held by IDG annually at the Yerba Buena Center in San Franciso—and many sought me out to write up their announcements.

Free software needed a lot of vocal support in the 1990s. It was still the victim of simplistic, biased impressions—and not just among those who dismissed it, but those who embraced it.

The arguments dismissing free software all too well-known and tedious. Potential users were reluctant to use free software because they assumed it would be low-quality, oblivious to the reality that the old tie between cost and quality has not only been severed but rendered meaningless in the open source world. These opponents are also afraid that open source software lacks support, even though there are people around the world who are expert in these projects and eager to offer their services, precisely because the software is freely available.

But I've had to recognize over the years that the free software movement nestled into some myths of its own—at least in the 1990s. Maybe the movement has wised up recently as the harsh realities of maintaining a community-driven project become clear. There is no clear way to fund free software, if you want it to be done correctly and kept current with changes in its environment.

>>>

Business models for free software have never been well understood. The creation of free software is clearly a powerful movement, and many industries tilt toward freedom as they mature. That's why, as I mentioned at the beginning of this chapter, many free software advocates assure us that the entire field of programming will end up where it began—as free software—before companies sold it in proprietary fashion.

But I have not seen anyone explain a coherent and comprehensive model for funding free software, a problem replicated with free culture. Eric Raymond, in his classic book *The Cathedral & the Bazaar*, laid out in 2001 some five ways to earn money doing free software, but none of them can be found in use today.

At the beginning, Richard Stallman held no expectations that free software could pay as well as commercial software. He seemed to think developers would choose lower rewards for releasing free software just because it was the right thing to do, as documented in Christopher Tozzi's book, *For Fun and Profit: A History of the Free and Open Source Software Revolution*. And this was potentially reasonable, because salaries for software developers were hefty and one could consider taking a pay cut to do things for free. But others, such as Raymond, thought that free software would somehow pay for itself.

Many companies profess an "open core" strategy, which involves a basic package of free software surrounded by proprietary extensions. It's an application of the "freemium" model described by Chris Anderson in WIRED

magazine many years ago. The freemium model works for many companies, and was a part of O'Reilly's strategy for decades. Although most of their content was for sale, they enticed potential customers to their site with free articles, and I spent much of my time writing them.

Open core makes sense in the abstract, but in practice rarely works. The open part appeals only to expert hackers who can get it up and running. If the core is really worth using, consultants outside the company can build on it and compete directly with the original company, as Monty Widenius's MariaDB and Percona's MySQL-based services compete with Oracle for its MySQL base. The MySQL ecosystem seems to be doing well, with all these companies tolerating each other's presence and collaborating on the core. But other open core companies fail to generate excitement.

Most free software seems rather to come from a trend I labeled "closed core" in a 2011 article. Here, a company offering proprietary Software as a Service, or some other product such as hardware, produces auxiliary software for non-core functions like performance monitoring or administration, and releases that software under a free license. By offering it for free, they develop an ecosystem of expertise and outside contributions.

Breaking economic incentives: Free licenses (1990s)

Although I already mentioned O'Reilly's brief fling with the documentation produced by free software teams—what I call community documentation—these writings have actually been intertwined with O'Reilly's publishing work from very early in the company's history. Their first big money-maker, the X Window System series, gobbled up material from the free documentation produced at MIT. And their biggest best-seller, *Programming Perl*, was written and updated in tandem with the Perl documentation. I myself had some success taking a free book from the Linux Community and producing a couple books from scratch that we published under free licenses.

These successes were uncharacteristic. Rarely does free documentation rise to the level where it's worth publishing commercially. If you dispute this criticism and point to books that moved from the free domain to the commercial domain, I'll narrow my assertion to this: Documentation rarely rises to the level where O'Reilly would publish it. I'll start with two examples of failure.

In the 1990s, when Perl was still king of the programming languages (some would call it a usurper) and our series was flying off the shelves, some editor laid an eye on the huge bramble of Perl libraries providing important programming functions for all kinds of tasks. Because the choice of libraries was so copious and their use was so complex, our editors got the idea of tidying up the free documentation and releasing it under our brand, as we had done with the X Window System documentation. The editor's hope was that a small investment of copy-editing would lend the material high enough quality to publish.

But the Perl documentation fell apart under this treatment, like rotting wood that one tries to nail up into a permanent structure. A small battalion of copy-editors jumped in to apply the clean-up techniques they had used with great success on drafts developed under the keen control of O'Reilly developmental editors. But the copy-editors found over and over again that their ministrations weren't working, because the source text was beyond repair. The ambiguities, missing facts, and sheer nonsense in the texts doomed the project.

Another stab at putting out community documentation sprang from a similar motivation: that of rounding out a successful series with a body of material too large and arcane for us to tackle in traditional publishing fashion. In this case, the series covered the MySQL database. After we formed a friendly relationship with the MySQL company, and collaborated on conferences with them, someone suggested we print their documentation as a reference manual. The material was in reasonable shape, unlike the earlier Perl documentation, but there was no compelling reason for people to shell out money for a printed version of the documentation, and it never sold an appreciable number of copies.

Now for a success story, one that came near the beginning of our Linux work. While seeking authors for this brand-new technology, I discovered a complete manual on networking by a GNU/Linux enthusiast named Olaf Kirch. A German who had never lived in an English-speaking country, Kirch wrote English that was idiomatic, highly informed, and even graced with humor. I fell in love with his free Linux Network Administrator's Guide and offered to edit and publish it, while leaving it under an open license that would let anybody republish it.

This was an expeditious way to launch an O'Reilly GNU/Linux series, and we made a good amount of money from the guide. But our experience also highlighted the risks of producing free and open books, as so many people—Cory Doctorow and Bradley Kuhn come to mind—have urged publishers to do.

While our book was in production, somebody scooped us by releasing a cheap edition with the exact text Kirch and I had carefully edited. When a marketing person at O'Reilly complained to me, I mumbled that our edition would have higher quality and would therefore push the interloper out of the market. But when I finally received my copy of our edition, I was horrified to find that the production team had omitted the first two paragraphs of the book. We sold enough copies to correct that error in the next printing, but I was embarrassed to have boasted of our high quality. It seems almost fated that this unique and egregious error would occur on this particular book, while it faced off against a direct competitor.

In the end, the competing edition disappeared, but not because of our edition's quality—just because ours benefitted from O'Reilly's marketing channels and reputation.

In general, I don't think that publishing documentation developed by a community is a winning business plan. O'Reilly has always done well by writing excellent documentation where the community's material was insufficient.

When I found an author to write one of our early Linux books, this time on the much more complex topic of how to code up a device driver, he asked for the book to be released under a free license. Subsequent authors have insisted that we honor this clause in the contract. (Because Linux Device Drivers was very successful for a long time, managers tried to scale back our commitment to the free license, and to offer merely an online PDF with no rights to alter and redistribute the book.) I never noticed anyone trying to upstage us with their own edition, the way someone briefly did on the Linux Network Administrator's Guide.

A final experiment with free documentation, this one also successful, came with our guide Using Samba. The free software in question was quickly becoming part of core network infrastructure. To explain the high stakes behind this book, I have to offer again a bit of computer history. You may find it rewarding as a behind-the-scenes glimpse at how computers work in most of our homes and offices.

»»

As computers became more and more connected during the 1980s, companies tried various schemes to make people in an organization feel like they were all connected in one gigantic system. The benighted Open Software Foundation and its DCE, which I have described elsewhere, revolved around this ideal. In particular, manufacturers wanted to give people access to files on their coworkers' local systems. Sun Microsystems had developed the universally adopted Network File System to do this, but it had design flaws and suffered from security weaknesses. So when Microsoft

> came to an understanding of networks and realized they needed local file-sharing, they chose a different technology called Server Message Block (SMB).
>
> An administrator in Australia named Andrew Tridgell, having found a need to share files between his GNU/Linux systems and Microsoft systems, dug out the use of SMB and decided to make file-sharing work through the extremely ambitious process of reverse engineering the protocol. He created a piece of free software called Samba that implemented a good chunk of SMB, and continued to chase upgrades and new features as Microsoft introduced them.
>
> Samba was so valuable that Apple Computers incorporated it into their Macintosh systems, which could make use of Samba because they were based on a form of Unix. In an updated form known as Common Internet File System (CIFS), the protocols used by Microsoft technology and Tridgell's matching free software are still the dominant way to share files on local networks.

I saw early on that Samba deserved a book, and found a good author. By the time we had finished the book and went to production, however, other publishers also had books in the works.

Encountering Tridgell at a conference, I asked him to write a foreword for our book, a form of endorsement. He in turn made an offhand comment that could be taken as nothing except an offer of a deal. "There are five books coming out soon about Samba," he said. "I wish one of the publishers would put one out under a free license so that I could endorse it."

I snapped up the offer instantly. But I had to persuade O'Reilly management that it would benefit us to gain Tridgell's stamp of approval, at the cost of allowing other people to update and release our book. We held a high-level summit of a dozen or so editors, run by Tim O'Reilly in our Sherman Street office. Even though we had already enjoyed success with the Linux Network Administrator's Guide, the going was tough. All the customary fears of opening up our own precious material to the public got tossed around for a long time.

But I had a powerful ally in this meeting: Mark Stone, who had recently joined us and was highly respected for his understanding of the free software community as well as the publishing industry. He calmed everybody down, indicated his unstinting support for my plan, and pushed it through. Our book Using Samba made us a mint and went through several editions. And no one put out a competing edition based on our text.

Birds of a feather: Conferences in free flight

Contents of this chapter

Something is happening: Dialog in Barcelona (2006)

At several points in this memoir, I have emphasized the value of conferences to O'Reilly, to the communities we served, and to my own growth. The best way I can demonstrate this value is a story from a conference I nearly didn't get to attend. And I certainly didn't know what the conference was until I got there.

My revelation had to do with the role of the author in modern media. Writing used to be a ponderous endeavor: the shuffling of foolscap sheets decorated by the carefully honed quills dipped into an inkwell that needed frequent refilling. When did writing get stripped down to telegraphic phrases? And did that metamorphosis elevate or degrade it? After seeing new media abused so thoroughly that they eviscerate democracy and the notion of truth itself, one would certainly say the latter. But I realize that different levels of writing—from the cursory to the exhaustive—benefit each other. I had to learn this from a children's book writer on a continent five thousand kilometers away.

I got to Barcelona through a friend I had made in my volunteer work for Computer Professionals for Social Responsibility, David Casacuberta. A Spaniard who moved to Catalonia but still preferred speaking Spanish to waiters decades later when I had dinner with him, Casacuberta was one of those academics who straddled computers and the social sciences. His research changed

from year to year, but generally concerned effective ways of applying computer science to social problems, and the effects computer science have on society. It's a perfect occupation to pursue in Barcelona, which for centuries has been on the technological leading edge and is determined to stay there.

Casacuberta, who was one of the few members of CPSR in Europe, first contacted me in 1998 to thank me for an article I published in The American Reporter about fascist propaganda. In this article, "Nazis, Neos, and Other Nasties On the Net", I boldly stated that we should not tolerate these perennial threats in our public places. Although I called for some forbearance online, because their postings did not create an immediate physical threat, I recognized that they could still be dangerous.

My balanced approach, although it certainly did not solve the problem (as we found later when the "alt-right" took over public discourse), was a contrast to the standard Internet activist view in the United States. Most American free speech advocates on the Internet simply said that anything goes, and that in a marketplace of free expression the truth will win out.

Europeans, having lived through Nazism, kept a more guarded view, and Casacuberta congratulated me on recognizing that there are many legitimate views on the subject of fascist speech. We kept up our correspondence over the years, and in 2006 he approached me with an intriguing request.

Barcelona hosts an offbeat conference called Kosmopolis—one I had never heard of, and whose existence I could scarcely imagine. Held at the historic Centre de Cultura Contemporània de Barcelona (CCCB), Kosmopolis treats of all things creative in all media. It has a special interest in what new technology offers the arts.

Casacuberta played some role in planning the 2006 conference, and was tasked with asking Tim O'Reilly to speak. Because he had no direct link to Tim but knew I worked at his company, Casacuberta asked me to convey the invitation to Tim.

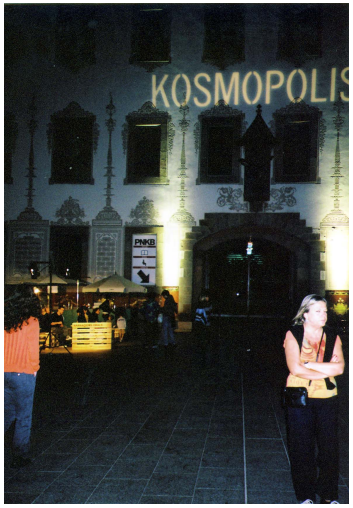
I lost no time blowing my own trumpet. I wrote back saying, "If Tim comes, he'll deliver the same business-oriented speech he gives at industry forums." I actually had no idea whether that was true, but sensed it to be. I continued, "If you bring me instead, I can give a talk tailored to the topics the audience is interested in." Casacuberta said he appreciated my insight, but that the other conference organizers wanted a "big name" that would draw more attendees, and for that reason had specifically demanded Tim.

My aggressiveness was matched by my luck that day. Tim thanked the organizers but said that he was already booked at the time of the conference. I was the second best choice, but Casacuberta said he would sign me up to keynote at Kosmopolis and even

admitted that I was the person he had wanted in the first place.

I haven't forgotten that I started this story as a lead-up to my revelation about writing on the Internet. But I must describe Kosmopolis itself first, because it was such a magical backdrop to a magical evening.

Kosmopolis runs on Iberian time, meaning that sessions begin at 6:00 PM each evening and go to about 11:00. That meant I could be a tourist all day and still attend the entire conference. Conference sessions included conventional talks like mine along with films and various kinds of performance art, such as a young black man who put together a rambling discussion on Colin Powell (former Secretary of State), tying together his lies in the service of the Iraq invasion with other aspects of his life, including a digression into possible incontinence that the speaker demonstrated on stage.



For my own keynote, I had prepared a summary of how I used the Internet for personal and work-related publicity. I could cite substantial experience here. I had published an article creatively exploiting the Web's distributed structure in the first issue of O'Reilly's early online journal, Global Network Navigator, which pioneered both advertising and edited content in 1993. In 1997, I had been highlighted in Fast Company magazine because I was staking out new ground by using the Web to promote my work. I was constantly looking for new ways to market O'Reilly's work on the Web in an age when social media hadn't caught on yet. (I didn't realize that being featured in Fast Company was anything to boast about until O'Reilly's publicist Sara Winge chided me for not letting her know so that she could capitalize on the achievement.)

Furthermore, I had just finished an article called "Characteristics of new media in the Internet age" that laid out a broad history of human creativity and divided it into three phrases. The first phase was the huge pre-printing era going back to the beginnings of consciousness, followed by an age characterized by strong copyright and the inviolability of the statement by a single genius, and now a totally new age that would be characterized by ever-changing works of art on which many people would collaborate. Sometimes, grand historical surveys like this appeal to me, even though I know they are crude and hide important disqualifying details.

All told, though, I don't think my talk was offering much new—not really worth traveling three thousand miles.

I was paired with a popular Peruvian author, Santiago Roncagliolo (a convenient pick because he was living in Barcelona), who had similar stories to tell about using online media.

The one truly clever moment in my talk was improvised. Just before going on stage, I met Roncagliolo, who let out that he doesn't like Bob Dylan, and that his readers sometimes harangue him for that. So in my talk, I added a short section about people being confused by the opportunities offered by the Internet, and ended with the comment, "It's like Bob Dylan said—you know something is happening but you don't know what it is."

How ironic that I should tease my fellow presenter that way! In truth I was describing myself. I didn't know what was happening, but over the next 24 hours I would find out.

A woman in her 30s stood up during the question-and-answer phase of the keynote and spoke passionately in Spanish (possibly Catalan—I don't remember). I did not understand, and some members of the audience tried to translate, but what they said was too fragmented to make sense.

But I had another public appearance at Kosmopolis: a workshop I held the next day. The same woman came again, tried to convey her comments again, and failed again to get through to me.

I saw her in the courtyard afterward, and in halting Spanish told her I regretted not being able to understand what she said. She responded, "Parlez-vous français?" I did indeed! We had finally found a common tongue. And everything flowed from that.

The woman was named Georgina Rôo and was born in Argentina, but had spent time in France before moving to Barcelona. She had written at least one children's book and was now offering photos and stories on an online media site. Because we didn't have laptops, we repaired to the speaker's lounge, where Kosmopolis had provided speakers with access to some large desktop systems. As other speakers crouched over computers or rushed meals, Rôo brought up her web site and explained her discoveries.

One of her favorite online activities was to post a photograph on which readers would comment. Often these comments would prompt her to take a fresh look at the picture and at her own story embodied in it, prompting ideas for new writing. In other words, for her the Web was much more than a site for publication and self-promotion. It created a dialog between her and her readers, and her readers helped to shape her content.

As I mentioned, social media was still a relatively new phenomenon in 2006. Friendster and MySpace had been offering forums for personal promotion for a while, but didn't seem to be changing the relationship of the individual to the collective. Facebook had been founded but was immature. What Rôo told me revolutionized my thinking about the creative potential of the Internet.

By now we felt like old friends. We had dinner together, and after the conference we connected on Facebook, where I could read her occasional Spanish-language postings. By the time I got back to Barcelona with family over a decade later, unfortunately, Rôo had returned to Argentina. But Casacuberta was still there and met me along with my wife for dinner.

Such is the magic of a conference. I flew out to Kosmopolis cock-sure of what I knew about authors on the Internet, and I returned with a new understanding of what I did not know.

Life at conference pace (1990s—2000s)

During the years when I attended two O'Reilly conferences per year—the Open Source Convention and the MySQL conference—as well as other interesting events, I cranked my gears up to the highest energy level and adopted a punishing schedule. Arriving in Portland, Oregon or Santa Clara, California on East Coast time, I would rise by five in the morning, sample the news (which during the early years could be found only in paper form), and have breakfast, which I often set up as a business meeting with a potential author or other valued contact.

This led—perhaps with a gap of a few minutes in which I could continue to catch up with news or with my email—to a full day of keynotes and sessions, punctuated with video interviews of leaders in the technical, business, or social aspects of the field, and with lunches scheduled months in advance among the key people with whom I wanted to deepen relationships and pursue future opportunities.

Nightfall brought only a different nuance to the treadmill of learning, networking, and celebration. I typically found a meal —sparkled up with alcohol—sponsored by a vendor or other organization that I wanted to draw closer to, and then went to several of the informal evening meetings that were held by a luscious variety of communities.

O'Reilly called these after-hours meetings "Birds of a Feather" sessions. We took this term adopted from Usenix, the classic gathering of researchers, system administrators, and community activists who circled around Unix. During these evening gatherings, the conference venue felt completely different. Daylight sessions teamed with people who rushed from one event or rendezvous to another, and who included groupies and hangers-on from marketing departments and the media. For the Birds of a Feather sessions, a kind of holy repose settled over the vast, open, featureless space. Coming upon a community discussion felt as if entering a sanctuary of the elect.

In Birds of a Feather sessions, speakers who had conducted formal sessions during the day could exchange views with their most dedicated followers in a less constrained moment. New projects that were too small to merit a formal conference session could present their vision and promise in order to alert technologists aiming for the next big thing in their field. Developers who exchanged ideas throughout the year over email and IRC (an early form of online chat) could sit side by side and lay plans for the upcoming year.

When the conference venue finally emptied out, at nine or ten at night, I would lug the laptop that had been fixed like a scabbard to my side all day back to my hotel room, and would whip out a blog posting in half an hour covering events of the day. After a final click putting the article up on the O'Reilly web site, I went to bed with a sense of fulfillment to prepare for a repeat of the routine the next day.

The interviews at our conferences provided an extra jolt of excitement as well as pressure. O'Reilly wanted to air interviews of five to ten minutes with industry leaders, as enticements to get more followers and encourage outsiders to come to our conferences. Sometimes I knew in advance some key attendees and could solicit a short interview on-site. Often, though, I heard interesting speakers or were introduced to people in the hallways or after-hours events and spontaneously asked them for interviews. I was certainly happy to interview major technology leaders, but also sought out voices that weren't usually heard, fostering diversity in race, gender, and geography as well as fringe projects offering promise.

In any case, scheduling was difficult. The video staff came for just one or two days with scattered fifteen-minute slots, and I had to coordinate the staff's openings with my own free time and that of the speakers.

I worked out many of the questions in advance, and my interviews always struck interesting points. Just one time I was disappointed, because of a poor decision made by the person I asked to interview. I had known someone in the open source health care movement (a rather tiny community) who got involved in a major UN-backed project. After we met in an open area at the Open Source Convention, he introduced his project to me with passion and insight, and I set up an interview.

Shortly before the interview was to take place, he called to say that he'd rather have one of the project's marketing people talk instead, and I gullibly agreed. The marketing person turned out to be all veneer—and a cheap veneer at that. Not only did he lack the historical depth and technical knowledge shown by the colleague I had invited originally, but he didn't even seem to know that such

things were important and of interest to an audience. This fiasco taught me to trust my own instincts when choosing speakers to interview.

For many years, the Open Source Convention gave shelter to another unique event, the Community Leadership Summit. This, like the Velocity conference, grew from a book that I proposed and edited. In this case, open source advocate Jono Bacon (more of a rock musician than a coder) had noticed that free software developers were grasping the importance of the communities in which they worked or volunteered, and also noted the challenges of maintaining a healthy and productive atmosphere. The term "community leader" was coming into use around this time, the mid-2000s, although I don't think it was yet honored with a job description and treated as a paid position. I believe that, at first, the role was taken on by dedicated community members, who forged its practices out of experimentation and tough experience. Bacon pitched the book *Art of Community* to us. He penned a wonderful exploration into many aspects of the field that sold extraordinarily well.

Bacon then proposed a conference that would piggy-back on the Open Source Convention. O'Reilly paid for a few rooms during the weekend preceding the convention, and several other companies stepped up to provide funding. But the amenities were frugal. As an example of this bare-bones organization, each person as they arrived wrote their own badge. In later years, I picked up on a trend permeating hip organizations and encouraged everyone to write their preferred pronouns on their badges. In this way I honored my own transgender child, while bringing us into the cultural shift adopted by other gatherings in the 2010 decade.

Volunteers checking in attendees to the Community Leadership Summit; Jono Bacon on the left

Everything else about the summit was pretty spartan, too. Lacking enough room for breakout sessions, we crowded about the round tables dotting the hallway and suffered from noise and distractions. Still, holding the conference in a set of rooms off the single hallway, which stretched in a wide leisurely circle with no corners to break one's view, brought us all into close association.

The summit served coffee and tea in the morning, assenting to the pumped-up catering charges required by the convention center, but asked people to scatter to nearby restaurants in Portland for lunch. The problem: There were no restaurants conveniently nearby (a puzzling dearth for a convention center that could gracefully host at least three large conferences simultaneously). The couple of sandwich places a few blocks away were soon overflowing with summit attendees, while some turned up their noses at these options and jumped on the light rail to savor the greater gastronomic opportunities downtown. Both sets of attendees were usually late getting back to the afternoon sessions.

But the spirit and quality of the Community Leadership Summits, which O'Reilly agreed to sponsor for many subsequent years, were electric. Who can network and exchange information more competently and entertainingly than community leaders?

The format was that of an "unconference", a forum driven from the bottom up. Neither sessions nor speakers were planned in advance. Instead, we trusted the expertise of the attendees, and allowed anyone to propose a session at the start of the day. People decided on the spur of the moment which sessions to attend.

The idea for an unconference was popularized by O'Reilly invitation-only events called FOOs (standing for Friends of O'Reilly), although I believe the idea of an unconference, and perhaps the term as well, predated the first FOO. At the Community Leadership Summit, Bacon bent the unconference ideal a bit by offering keynotes as other conferences did. Toward the end he adopted a more standard conference format.

Topics at each summit were as varied as the attendees. The summits were so intriguing and unprecedented that they attracted people not associated with free software. I was amazed to find people who flew long distances for the Community Leadership Summit without even attending the Open Source Convention that followed. The summit had an audience and a community of its own.

As an example of the bonds one could form, one activist in free software named Jeffrey Osier-Mixon met with me just once a year when we both attended the summit. I believe our very first meeting took place during the excitement of the initial summit, perhaps at the very beginning of the day. Many lively conversations followed. We had no particular project to work on together, and rarely exchanged messages during the rest of the year, but re-established the relationship at each annual summit. Our conversations were of the type where a couple words and cocked head could convey shared values and a common understanding. Friendly, cheerful,

and always ready with a smile, Osier-Mixon gave off the sincerity of someone who cared deeply for causes. Many years have passed since we talked, although we're connected on LinkedIn.

I offered two sessions regularly at each CLS. The main one brought up my obsession with how to produce documentation by volunteers within free software communities. This session was related to the research topic I informally pursued for many years and even thought of making into a new career. My forum was always well attended and drew people who knew more than I did about producing such documentation.

Regarding documentation, incidentally, I was one of the people who assiduously posted minutes from the CLS sessions I had attended, so that people anywhere in the world could get some benefit from our discussions.

The other topic I spoke on a few times, including a keynote with slides at one CLS, concerned lessons that organizers at the conference could learn from classic community organizing—the 1940s movement started by Saul Alinsky and still embodied in the Industrial Areas Foundation. I explained my own growth as an organizer and some of our successes.

The summits return to my mind wrapped in an aura of emanating potential. Although plunked down in an uninspiring physical setting, with cavernous hallways and windows looking out at highways, each assembly of volunteers representing far-flung communities and places built a place of acceptance, mutual support, eager inquiry, and even mutual love. I think that a bit of the spirit of Portland, a depository for searchers and misfits, contributed.

Attendance at the summits, however, extended my conference participation to the point where it stretched my stamina past the breaking point. Two days of Community Leadership Summit plus five days of Open Source Convention were too much for my constitution. Around the fourth day of one Open Source Convention, I admitted in my blog post that I took off an hour "to get a more horizontal view on things."

Another time, poor planning brought me low. I held back on food consumption through the whole day, anticipating the hors d'oeuvre to be served at an evening event. But I was keeping somewhat kosher (an observance which I've adopted and abandoned at different points in my adult life), and saw at the event that everything had bacon, shellfish, or some other traif component. So I refrained from eating, but partook freely of the alcohol. I went briefly to another conference party but wisely left the liquor alone, realizing that I'd overdone it at the first party.

Dawn brought with it a terrible headache, and I stayed in bed till lunchtime. I then dragged myself down to the show floor and plopped myself into a seat in the O'Reilly booth. Jono Bacon came by. I pointed to a professional actor who, cavorting as part of a vendor promotion, was walking around the show floor fully covered in a bunny suit. I told Bacon, "I know I'm suffering from a hangover, but this is ridiculous."

Alcohol causes a lot of evil at professional conferences (contributing to dangerous behaviors such as sexual assaults) but remains a key element of connection and community celebration. I heard one person riff on the popular term "peer-to-peer" to claim that the most productive developments at conferences occur "beer-to-beer". At the Open Source Convention and MySQL conference, insiders could sample the unique licorice-flavored black vodka brewed by Monty Widenius, the creator of MySQL, at either a Birds of a Feather session or a private party.

For many years, one of my most common type of dream locates me at some conference—never a real-life one, but some fictional conference meeting about some cloudy notion. This conference is always held in an intriguing and mentally stimulating space very different from the desolate venues for most real-life conferences. I think that these recurring dreams are concocted from the actual experiences of personal connection, enlightenment, and joy I have had throughout my career at conferences.

FOO conferences were special because of the extremely limited invitation list. People occasionally complained that they were invited to one FOO but passed over the following year, but that's because the whole point was to rotate experts. There was a sort of natural ceiling to the number of people who can construct a temporary, weekend-long community—about a hundred and twenty, reminiscent of Dunbar's number.

FOO pushed the boundaries of behavior physically and mentally. I minimized sleep at these events, because one could pop in on interesting conversations until two or three each morning, and productive interactions started again around six. Although some attendees literally camped out at the O'Reilly campus in Sebastopol, I just found a quiet spot in one of the buildings, bedding down with my backpack as pillow and my jacket as blanket.

A few attendees would retreat to the comfort of local hotels at night, but most of us didn't want to miss a minute. After all, we were constantly rubbing shoulders with world leaders in computing, science, education, and other fields. A few health FOOs were held during O'Reilly's period of interest in health IT, either at the Sebastopol campus or at the conference space offered on the

tenth floor of Microsoft Research in Cambridge, Massachusetts. Both spaces offered nooks where people could meet quietly for brainstorming.

But that kind of untrammeled headiness can turn bad. For me, FOO conferences were kaleidoscopes of wonder. But I heard later that various forms of licentiousness took place, including assaults. I never saw those things myself.

Raised to be frugal, I always took note of the expense of conference travel. I can't remember ever crossing the country on a non-stop flight for a conference, because I always looked for a cheaper option. I wouldn't reserve a room in the block reserved by O'Reilly, unless they encouraged me to fill a block they had reserved in advance. I enjoyed the task of finding a budget hotel within walking distance of the conference center. And one manager, Frank Willison, expressed his appreciation when I told him that I was staying at a cheap hotel four blocks from the conference site and carrying a box of books there by hand. (That was not for an O'Reilly convention, but for an Ottawa Linux Symposium, a highly sought-out place to watch the sprouting of new kernel development.)

My son, as a teenager, was trying once to get his head around the concept of an expense report and the trust that managers must grant the employees who claim reimbursements. Have I ever submitted false expenses, he asked?

This is one of those moments when a parent can help shape his children's approach to life. Although expenses require receipts, that is scant protection against skimming off the top, because profligacy can be disguised or receipts padded.

With my son, my goal was not just to demonstrate proper morals but to ground them in good sense. So I said that I try to minimize expenses, and never pad them. This is not just because doing so would be unethical, I explained, but because every dollar I take for myself is a dollar that my company can't invest in its growth. And I figure that a successful and expanding company will benefit me more in the long run than charging an unnecessary restaurant meal.

My attitude loosened a bit after one Open Source Convention in Portland, Oregon. I saw a colleague from Washington, DC at the site the day before, and asked him whether he was staying for the conference. No, he told me, O'Reilly had flown him out just for that day to attend a meeting. I decided that I did not have to show any more concern for travel expenses than my managers did. Nonetheless, I did not start frequenting three-star Michelin restaurants, or even fly non-stop.

There was no need to credit altruism in the way I put O'Reilly's needs first when talking to my son. I always identified the company's financial success with my own. You have seen several examples of this in the memoir. Here's another.

During the mid-1990s, when editors were high priests who directed every turn taken by the company and lapped up a sizable portion of the resulting bounty, we hit a rough spot financially, perhaps an augur of the crisis to come in 2001. I wrote to Tim O'Reilly to say that I would be willing to give up some of my editor's income to help other parts of the company. The situation must have been dire, because I got a phone call from Tim where he said with a nervous chuckle, "Andy, I think I'll take you up on your offer." All the editors had to make a sacrifice to keep the company afloat. It may be a sign of privilege to make a personal sacrifice for the greater good, but recent world history suggests that privilege has not bestowed that quality on most of the wealthy "one percent". For some reason, it seemed instilled in my bones.

Tremors of upheavals to come: O'Reilly launches a conference (1990s—2000s)

Stereotypes make it easy for people to pigeon-hole issues on which they don't want to expend thought. One way the average person has tried to deal with the importance and financial success of computer programmers in our lifetimes is to adopt the slur that programmers, perhaps congenitally, lack the skills for social interaction. I have discovered a full range of personality types among programmers and other computer professionals. Some, it is true, show impatience with the small talk about pets and traffic that populates many everyday conversations. They prefer to seek out people who can speak at their intellectual level about things they like to do. But the craving for community is strong, and spurs them to cross oceans for conferences many thousands of miles away. There, they can go from dawn until late in the wee hours connecting with people of like minds.

Certainly, I've encountered some curious types among the computer set. The Perl community seemed noted for odd personalities who often developed rivalries verging on feuds. On many mailing lists across the computer field, a perennial topic was how to deal with "toxic members" or just with routine insensitivity.

One leading programmer, a young man with serious thin lips, happened to sit next to me in one of the rooms set aside at computer conferences for attendees to take time out and conduct personal work. This room contained a couple dozen hackers wrapped up with their laptops, so the atmosphere was rather unsocial to start with. This may have colored the programmer's mood

as he noticed that his laptop was low on battery power, glanced over to see that my laptop took up the nearby wall socket, and then poked his nose at my screen to determine that my laptop had twice as much power as his.

A normal person might have said, “Excuse me. I notice that your battery has a lot more charge than mine, and I need to do some important work before my laptop loses its power. May I use the socket you’re using?”

But this is what he did instead: Abruptly pulling my plug from the socket, he said as if citing a rule from some community guidebook, “I have only 16% battery power and you have 34%, so my computer takes precedence.”

Rudeness can also be an emotional adaptation to chronic disappointment. Those of us who bumbled our way into the computer field while young have weathered several generations of that fast-changing industry, a pace that can bewilder and all too often sideline its leading members.

Thus, at one Open Source Convention, I ran into a founder of the early social media company Friendster. She expressed a good deal of resentment, which I attribute to her entering the field too early and being forced to stand by as a witness while MySpace and then Facebook reaped unconscionable amounts of money using the concepts and even the terminology her company pioneered. She channeled her anger into the lack of recognition women suffer from in computing, which was and remains an important critique. I tried to mollify her by mentioning that I had written up an interview I had with Margo Seltzer, an important manager who emerged from the BSD community and eventually became an Associate Dean at Harvard University. The Friendster founder snapped back, “Who gives a fuck about Harvard? I started a company!”

The occasional irksome encounter at a conference—which under-represented groups no doubt suffer from more than I do—provides mostly spice for the many unexpected meetings that reveal an unexpected common humanity. For instance, I sought a meeting at one health IT conference with the founder of a small company. He told me that he was excited to meet me because he had checked my Twitter feed after I made the inquiry, and found a tweet that won him over.

What was this tweet? Some insight into technology, some apt observation on the health care field? No. What happened is that, as part of my research into big data, I had found an Israeli app that tracks Arab terror attacks and presents them on a map. I wrote that I would approve of such an app if it included Jewish terror attacks. The company founder, a Palestinian, saw in me a sympathetic observer he could talk to.

So there was no finer opportunity than a good conference to meet new people. Even later, in an age of high-speed telecommunications, conferences offered the highest bandwidth by far for information exchange. Everyone who was anyone had to attend. Right up to the COVID-19 clamp-down on meetings in person, conferences provided a critical social and community-building function.

Everyone understands that what happens outside the session halls is the most important experience, although many fine conference sessions have inspired people as well. I have explained repeatedly in this memoir that conferences were the focal point of O'Reilly's community-building and its research. I hope that the personal stories I recount in this chapter will help you see the importance of conferences, if you haven't experienced it yourself.

The greatest conferences bring together the leaders of movements. Debates are saved up for discussion at the annual conference, where decisions are made that drive the field for years to come.

Can a conference achieve even more? Yes, it can make a statement that fans out across the community and the general public, opening eyes to a trend that had been previously overlooked. Perhaps it can augur the future.

O'Reilly's Perl conference did all these things. One of our marketing managers decided to take on the steep challenge of putting on a conference, with no prior experience, because the company could see Perl coming to occupy a unique and central role in computing that fell outside the horizon of the trade press. O'Reilly sensed that we had a story to tell that would benefit our product line while giving insight to managers and users throughout the computer field. At the same time, the Perl Conference gave Perl programmers and hackers of all persuasions a place to gather, as well as a platform for Perl creator Larry Wall. Small conferences about Perl had been cobbled together by volunteer enthusiasts before, but they were easy to ignore outside the community. An O'Reilly Perl conference was news.

We also took risks in program choices. Many conferences bring in a keynoter who does not address the theme of the conference directly, but proffers an expert viewpoint from another field that attendees will find relevant. (I've delivered a couple such keynotes myself, about peer-to-peer and about government adoption of open source.) Going further, O'Reilly's conference tradition included—along with the usual corporate pitches that sponsors insist on—keynotes that expanded the minds of participants more than conventional keynotes. The prototype for this approach, I think, was a keynote Andrew Schulman delivered at the first Perl conference.

Schulman was a Windows programmer with no knowledge of Perl at that time. He had recently joined O'Reilly to help us expand our reach and add some sparkle to our editorial message. His keynote at our conference had nothing to do with Windows or Perl. Instead, he helped the audience think about the evolution of the Web, which was moving from static content to dynamic pages generated upon demand. I particularly remember his showing how FedEx (a legacy company that seems to be doing fine, even though more and more documents are taking digital paths to their recipients) used the Web as a doorway to its database and the links (URLs) as a programming tool.

⋙ Every single FedEx package had its own web page, with what Schulman called "the URL from hell". But the page didn't exist until the customer clicked on the link. What happened next was described by Schulman in his review of this memoir: The incredibly long URL "invokes a process on Fedex's server to dig into Fedex's database in real time and turn the information there into a web page on the fly. My larger point was that URLs are a way of doing remote computing, i.e., invoking processes on other systems. The programmer and author Jon Udell was making this same point at around this time in articles in Byte magazine." A pretty nifty idea to air in 1997, when few were talking about URLs as universal identifiers and about the Web as a vast database recording the world's transactions.

In retrospect, our sponsorship of a Perl conference heralded the beginning of a shift in computing so transformational that the term "tectonic" would be understating the case.

⋙ Perl was becoming popular because system administrators were becoming programmers. This was the glimmer of the DevOps movement that matured and found itself a firm place in corporate development a good 20 years later. Programmers and their companies started using the term DevOps in the 2010 decade as they noticed the traditional walls breaking down between the development teams and the operations teams who were responsible for getting the programs into everyday production. Operations were getting automated, looking more and more like the programming that developers did.

The most imperative driver of modern system administration was the speed-up of social and business change, which since the 1990s whipped companies to create and upgrade applications at an unprecedented pace. Software came to undergird more and more businesses, and these businesses became dizzy over its potential speed of change, as at Facebook with its notorious slogan, "Move fast and break things." Managers pressed for changes by the hour or minute for all kinds of goals: to respond to threats, to fix bugs, to try new features on selected groups of guinea pig

visitors (A/B testing and bandit testing), and to roll out permanent changes. The impatience of the mobile consumer was matched by the impatience of the corporate planner.

Programmers could not achieve that pace through the traditional process: development first, then testing by another team, and finally deployment by yet another. Each stage introduced friction, required its own set of tools, and made one team dependent on another possibly unreliable team to meet its deadlines.

The logjam was broken by new tools for automating the steps in testing and operations. A new approach to connecting programs called RESTful APIs, which ran over the Web like the peer-to-peer services I have described elsewhere, provided a lightweight interface that programmers could easily learn and that every programming language could support. This in turn led to the unification of all development and deployment. It seems inevitable that these trends would make Software as a Service more and more attractive, because no one wanted to distribute new software to users at that pace. We have come back to the factors driving cloud computing that forced their way into the beginning of this memoir.

The merger of programming with system administration represented by DevOps wasn't happening at all in 1997 when O'Reilly launched the Perl conference. But the groundwork was being laid. System administrators were dealing with larger and larger installations of more and more complex systems connected in multi-layered networks—and failing more often in ways that were more difficult to fix. System administrators thus needed automation, and they learned to program so they could have eyes everywhere to monitor everything and could relieve themselves of mundane fixes and other tasks. I see that as a key step toward DevOps.

We didn't say all that at the Perl Conference. But we helped prepare the table for the feast. Also like other fine conferences, this one covered a wide range of ideas beyond the Perl language itself, as demonstrated by the keynote I mentioned by Andrew Schulman. According to one record, Eric Raymond also gave a speech at the conference based on his ground-breaking article about free software and community-based innovation, *The Cathedral & the Bazaar*.

Logistically, the conference was also a tremendous success, a tribute to the marketing team who took it on without experience in the field. We put on a second Perl conference next year and then expanded the scope of the conference even further, calling it an Open Source Convention and covering whatever was on the minds of programmers using free and open source software.

I remember one year that the different programming languages we covered were awkwardly segregated—which does not engender healthy conference interactions—each being assigned a different hallway in locations scattered through the hotel hosting the conference. I felt a chilly sadness walking through each hall, as if entering the vault of a forgotten archive. When we grew large enough to occupy a more standard convention center, whether in Portland or in San Jose, California, all tracks were physically integrated.

Strange eddy in the river of the Web: Windows Live (mid 1990s)

I've had the good luck to receive a couple conference invitations through authors who fell behind in their writing and felt guilty. They apologized for their missed deadlines by giving me the opportunity to travel and participate in some curious events. These authors didn't realize that many other authors behaved far more irresponsibly, and perhaps didn't see past my personal relationship with them to understand how their failure to produce their books affected O'Reilly's revenue and product line-up.

Anyway, in the mid-1900s I received an unusual invitation through an author who worked at Microsoft. A major new web service called Windows Live was in gestation, so the company was convening a couple dozen people they considered leaders on the Web to review early plans. I don't think I had anywhere near the knowledge and depth of experience displayed by the other exalted attendees, but my author wangled me a place on the team.

It was a very congenial group of people to caucus and dine with. Microsoft paid for everything: our trips to Seattle, hotels, meals, and a gift certificate worth over $100 that we could spend in their campus store on our last day before flying home.

We were all hackers at heart. We would have been happy staying at a motel and having a burger for dinner, but Microsoft put us up at the prestigious W hotel and brought us to toney restaurants for hors d'oeuvre and fine meals.

The W was quite an experience. I don't remember what the room was like, probably because I'm normally oblivious to my surroundings, but I did notice that each floor had, opposite the elevators, beautiful eighteenth-century architectural sketches. They were originals, I think. I have always loved galleries, and the elevator landings of the W Hotel formed a 24-story gallery. I took the elevator up and down one day, floor by floor, to check out different sketches in four or five hallways.

I was also in the W lobby one day for a strange sonic experience. They piped in techno music, which I suppose was supposed to flaunt hipness while establishing a relaxed ambiance. At one moment, I heard the techno fade out and the famous Adagietto of Mahler's fifth symphony come on. After ten minutes, the Adagietto finished and the techno resumed.

But let's get to the point of the meeting. What was Windows Live? To understand it, you have to put yourself back into the 1990s, to think of who dominated the Web at that time, and to grasp the size of the transition Microsoft was going through.

>>>

The Web was started by scrappy independent writers, resolutely democratic and decentralized, but as the Internet revolution brought millions of people online and the Web became popular, large companies angled to take it over. CompuServe and America Online, which had prospered through private networks, sought to create similar circumscribed settings on the Internet. Like Apple, which created its iPhone store to let outsiders participate under strict controls, these big companies were afraid of the open Internet, sensed that part of the public was also afraid, and wanted to provide a feeling of community and connectedness while keeping the companies as benevolent if slightly fusty chaperones. Critics called these offerings "walled gardens". By the time Microsoft realized that the Web was a big deal and that they had to be there, Yahoo! had become wildly successful by offering weather reports, movie reviews, and other detritus on a site called a "web portal".

So basically, Windows Live was meant to be Microsoft's web portal. To our motley team of reviewers, Microsoft presented a set of mocked-up screens and talked about how the visitor would navigate through them, a process we now call User Experience or UX. The screens and UX were pretty unimpressive, and we all gave candid critiques.

After the sessions and meals and hallways conversations, we had our shining moment in the Microsoft store. Some of the team members were like kids in a toy shop, snapping up things like whatever game console Microsoft was pushing at the moment. Still the big free software advocate, I looked around and saw hardly anything of interest, so I saved my coupon and handed it over to the editor back home at O'Reilly who was in charge of our Microsoft book program. What I still preserve from this adventure is a black nylon/cotton jacket saying "Windows Live" on the back, appropriate to pull out on the first crisp Autumn days each year.

I believe Microsoft eventually did launch Windows Live, and it went across about as well as the Microsoft store did with me. I don't think Microsoft really developed much of a Web strategy until it mimicked Amazon.com's successful Amazon Web Services with a Microsoft cloud offering called Azure. By this time in the 2000s, the various large companies were implementing the walled garden concept once again, this time as cloud services for programmers and data scientists. And that harks back to the very beginning of this memoir. But I have one more set of memories to share before wrapping up.

☞ *Opening the book: The life at O'Reilly*

Opening the book: The life at O'Reilly

Contents of this chapter

"Honesty...let us work on it with all our malice and love."
—Nietzsche

The lonely office and working remotely (2000s)

This memoir has largely been about interactions: some connecting across thousands of miles, others occurring day after day in a single office. Rich interactions took place in the first office I came to, the eccentric Brickyard on Sherman Street, where the bulk of the stories in this chapter take place. By the time of the big layoff, the East Coast office of O'Reilly was in Downtown Crossing, Boston, but I never had a desk there. I was working remotely by then. I did my best to keep strong relationships with other employees, forged over years of close collaboration, but these ties gradually frayed at the building we occupied before Downtown

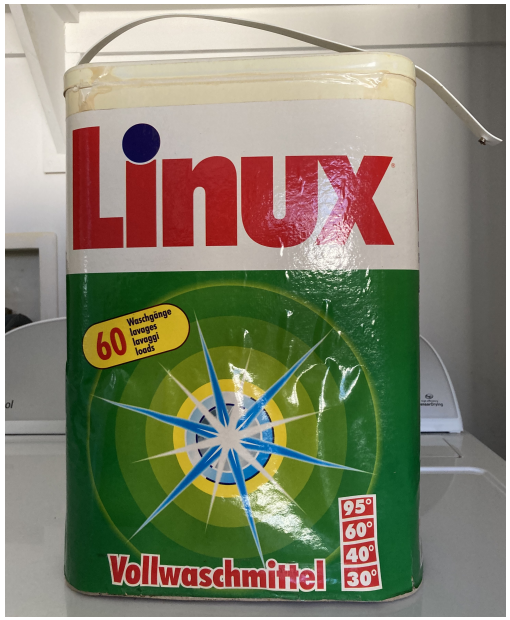
Crossing: a conventional office complex on Fawcett Street, on the western side of Cambridge, to which we moved in the early 2000s. The culture there eventually whittled away any motivation I had to come into the office.

I had nothing against Fawcett Street. The setup was cozy for me, as I lived in the next town over. A ten-minute bus ride would take me to work (twenty minutes perhaps, when Cambridge's highly resented rush-hour traffic choked the main roads), but usually I walked. My fifty-minute walking route took me through a forest and by a pond. After the sounds of the woods receded, I inserted earplugs and listened to MP3 files that I had ripped from CDs. I then traversed a suburban landscape that gradually grew more crowded until I was right across from the woods of Fresh Pond park.

In the winter, I put on snow grips and bypassed the forest. Although a few years before I had broken my ankle falling on ice, I overcame my fear of the winter weather. The only time the commute was difficult was when the city of Cambridge tore up Concord Avenue for years on end right next to our building, and eliminated the sidewalks on both sides. I can now issue my final piece of advice in this memoir: Don't make a habit of dodging Cambridge automobile traffic.

Another nice thing about Fawcett Street was my personal office, an office space I loved for a job I did not like. I associate the space now with the mid-2000s slump I have described elsewhere. But I enjoyed the office's repose. It was tucked into the corner of the building, right opposite the stairs I used to access the office (I climbed the four flights for exercise) and far removed from most of the bustle on the floor.

I decorated the walls of this office with memorabilia that held meaning to me, including a painting by Judy, and displayed oddities I had picked up over the years, notably an enormous empty soapbox bearing the name "Linux". (Yes, a German company had irreverently decided to use the name to sell soap, managing to sidestep trademark issues. They earned worldwide fame for this marketing ruse, even though nobody bought the product. Germans are very particular about their soap, and my friends in our Köln office told me that the Linux soap had earned a poor reputation.) I could also display a two-foot-high glass penguin in bilious yellow that I won at a trivia contest at the 2004 LinuxWorld conference.


Linux
60 Waschgänge
lavages
lavaggi
loads
95°
60°
40°
30°
Vollwaschmittel

The office was rounded out by a gawky desktop computer running the GNU/Linux distribution of the day—recalcitrantly bending to my will when I exerted enough mental force—and a copious bookshelf collecting translations of my edited works into languages I could read and those I could not, old books from O'Reilly and other publishers that I had found useful at one time or another, journals, and other mostly worthless historical artifacts.

The only drawback of my office was that the window faced north onto Fawcett Street instead of south toward the lovely trees of Fresh Pond. The view of parking lots and boxy buildings was not aesthetically notable but historically inspiring, because I could see right onto one of the old buildings of Bolt Beranek & Newman, later after a break-up to have the name Genuity slapped on before the whole edifice was finally replaced by a more modern structure. I could swell with excitement at being a few yards from the ground where stood the company that designed the Internet, after performing other equally critical technical roles during World War II.

The challenge of moving to Fawcett street was that of keeping our minds supple and creative while leaving behind the funkiness of Sherman Street, which I compare to the atmosphere of a college dorm. Certainly, the new space conformed more to the look professionals expect an office to have. We didn't have to deal with leaks from the ceiling anymore, and our electricity use no longer had to beg the wiring for cooperation. This was probably a necessary move.

However, the building at Fawcett Street had its own little lapses. You needed a card to enter the employee garage, which in our popular Cambridge location was a worthwhile control measure—but why did you also need a card to leave the garage? This made things difficult for visitors and security staff.

And then the smart lights. The building's management thought it would be a great energy-saving measure (such things are perennially of interest in the city of Cambridge) to place sensors that turned overhead lights on when people enter the room and thriftily shut them off when the room was empty. But the sensors were merely motion detectors, by no means designed for the purpose to which the building put them. If you worked quietly for five or ten minutes, the lack of motion convinced the light that the room was empty, and suddenly you couldn't see anything but the glowing computer screen. I saw one employee develop an automatic habit of nodding her head every few minutes, in order to trigger the motion detector and preserve the light. Luckily, it took only a couple months for management to disable the feature.

Building security was equally crude. A guard would open the building sometime in the morning and lock it in the evening. There was no way to get in when the guard was gone. (I suppose that occupants had card access, but there was no accommodation for visitors.) So one frigid winter day—when the temperature was much lower than even what we were accustomed to in New England—I found my colleague Tim Allwine, from our California office, waiting outside the door, inadequately cloaked against the biting wind. He had arrived for a morning meeting, and was rewarded for his promptness by being abandoned to the elements.

I don't remember any successful cyberattacks on our web site or internal systems during my 28 years at O'Reilly. I may not have been informed if one occurred. But in the Fawcett Street office, one breach was impossible for anyone to ignore.

One day, all the printers started spewing sheet after sheet of garbage text. The administrators ran around shutting them down, then investigated what was going on. They realized that some employee had opened an email attachment with malware. After installing itself on the employee's computer, the malware copied itself to every other computer on the local area network, where presumably the malware could launch further attacks. This attack vector suggests that the malware's creator meant the malware to remain hidden, in which case the attacker made a serious mistake. The malware could not tell the difference between desktop systems and printers. When it copied itself to all the printers, they took it as a file to print and thus emitted whatever printable characters corresponded to the bytes in the malware's binary file.

At some point our management succumbed to the mania for open seating, which had been growing in popularity for some time. I spent most of a decade in cubicles before coming to O'Reilly, and could appreciate the desire to leave behind those indignities. To tell the truth, I could work anywhere and could put all distractions behind me.

Open seating has spawned many debates. It does seem to increase collaboration, while tormenting people who need to concentrate in peace. My impression is that the open seating movement was started by designers in places such as advertising agencies and architecture firms. Here, creativity flourishes in the mingling of many ideas. In fact, before O'Reilly moved to Fawcett Street, the four or five members of our design team took over an open space at Sherman Street and made it their design cave. Because design took on an aura of indispensability during the 1990s and 2000s, leading entire cities such as Barcelona to rebrand themselves as design hubs, I believe that a mania for open seating spread from these creative types to the general office environment where its suitability was much more dubious. In any case, this was the direction O'Reilly took.

Following a plan put together by office manager Rita Scordamalgi and others, we were given some time to pack up. Being told that shelf space would be available, I culled my belongings down to four boxes, mostly consisting of the superannuated books built up over the decades.

The office closed, reopened. When I returned, I found the four boxes next to my mercilessly exposed desk, but none of the promised shelving. So I rid myself of the little sentimentality I had maintained for old translations and other obsolete relics of my work. I offered the books to our donation program (where we send old editions of our books to communities that can't afford the new editions but can benefit from the preserved versions of the past) and sent my swag to O'Reilly's archive. I took home a couple boxes of such things such as the overflowing collection of paper clips and rubber bands that I felt too useful to go into the trash, and the few books that held signatures or other meaning for me. The yellow penguin and the soapbox bearing the name LINUX went into my basement.

I was the only person using GNU/Linux in the Boston-area office, although I imagine that some customer support personnel or other techies in the Sebastopol office were using it in some capacity. I was on my own, and the environment proved unforgiving as the GNU/Linux communities strained to keep up with hardware manufacturers and commercial software developers unsympathetic to the operating system. After I installed an early distribution of GNU/Linux, one setting allowed networking while refusing to support graphics, and the other possible setting did the reverse. I forget how I got both networking and graphics to work.

After tiring of making it a second career to tinker with GNU/Linux installations, I took a tip from manager Mike Hendrickson and bought a laptop from Intel with GNU/Linux preinstalled, a new offering from the company. I immediately fell into a black hole in their support, after a routine update to the Ubuntu GNU/Linux operating system shut down the laptop. Intel took absolutely no responsibility for the product they had been happy to sell me, and directed me to Ubuntu forums, where obviously no one knew anything about the Intel installation. I had to wait for another routine update that fixed the problem as mysteriously as it had fallen on me in the first place.

So by the mid-2000s I was ready to give up the pride of ownership in a GNU/Linux system and to take a MacBook from my system administrator.

The other fixture of my time at Fawcett Street was my health club, Mike's Gym, which I was also ready to give up. There was a time when I would make the 50-minute walk to work, spend an hour at the health club on the stationary bike and weight-lifting

equipment, then walk back home. The social environment at the club was also stimulating. I could talk to other men about technology while standing around nude in the locker room. I remember one person asking me, as I was preparing for the shower, "What did you think of the tickle conference?" Luckily, I realized he was referring to a T-shirt I had worn bearing the name Tcl, an obsolete scripting language which was properly pronounced "tickle" and presented an undeserved challenge to Perl for a few years before both were supplanted by the Ruby and Python languages.

Harvard professor Henry Louis Gates Jr. also frequented Mike's Gym for a while. I suppose that this illustrious scholar, like me, was seeking a cheap exercise option. We chatted and I offered him my business card. I learned that he was dealing with serious leg and lower body problems, although I did not realize until I read his memoir *Colored People* later that he had suffered a serious and long-undetected injury called a slipped epiphysis in childhood. I would have talked to him about it had I known during our Mike's Gym sojourn, because it so happens that my own child Sonny had experienced the same problem and nearly suffered permanent damage during the several weeks during which we, too, didn't recognize the problem.

Basically, I liked the Fawcett Street location, even after I had to give up my personal office and take a seat on the floor with forty other staff. But I was rarely working day-to-day with these people.

When our artist worked in the Cambridge office, I found it valuable to meet with him and go over figures together. But he shifted into video production and the figure-drawing function moved to California.

Book production also became more standardized and the design elements became fixed. The production managers discouraged one-off, idiosyncratic designs. So there was rarely a need to meet with the production and design staff, as I often had in the past. The regular weekly meetings I had been forced to attend with my manager or a production manager, luckily, also ceased. These meetings raised problems none of us could solve. Eventually, someone realized that chewing me out over deadlines missed by authors was not bringing in books any faster.

I have already described how we gave up an intriguing, quirky Sherman Street space for the more dependable office environment of Fawcett Street. I thought that a lot of the intriguing aspects of working at the Sherman Street location got left in its dust as well. The new office sported a few enhancements, such as a little game area (also available later in Downtown Crossing) and for a while a 3D printer. But the community wasn't the same.

There were the requisite holiday parties, but no quirky gatherings such as when an editor visited Italy, wrote a hilarious skit about his experiences, and divided the parts among us to perform in the Sherman Street lobby. Nor do I think the office would come together as the Sherman Street office did on September 11, 2001 as the World Trade Center in New York City fell: A number of us gathered in the lobby holding hands and singing songs of peace from our various religious traditions.

Some of that communal atmosphere had vanished along with Frank Willison; most of the rest failed to find a chink through which to pass the Fawcett Street walls. I felt with some distress the loss of the old culture, and there seemed no way to recapture it.

The final straw came when I entered a conference room for one of our remaining weekly meetings and found myself alone. Everyone else, including a few people right across the open seating area (remember that this was supposed to increase collaboration), chose to phone in. I suspect that people preferred to phone in so that nobody could see that they were working or reading email on their computers throughout the meeting. But I think they also demonstrated a disregard for the communal vitality of a colocated staff.

So I decided I could work from home. The only thing I missed was my gym, but I created my own work-out routine using weights that I could wield in my basement. I let my membership lapse just a few months before the club closed for good—not, I aver, because I abandoned them but because management was idiosyncratic and unpleasant. Many people disliked the owner, although he somehow took a liking to me and helped me somewhat during my stay.

As another historical tidbit, I found that walking long distances became less fun because I could no longer rip music from my CDs to create MP3 files. That change came when Apple introduced its iPod device and its own music service. Apple simultaneously "upgraded" its iTunes product to eliminate the CD-to-MP3 feature. For a long time, after introducing iTunes, Apple had heavily advertised the ability to create MP3 files or your own CDs through a slogan "Rip, Mix, Burn". But when their new business turned "Rip, Mix, Burn" into competition, they simply made it impossible to do on their computers. Thus I encountered in my daily life the tyranny of non-free software that I had been documenting in my articles.

The question of remote work has popped up at several points during this memoir. I've highlighted problems in empathy and communication that can come about when employees are separated, and have rejoiced in my own experiences sharing an office. But I've also calculated the heavy costs of concentrating people in one location. O'Reilly has always worked with people from afar, and

even before an obscure bat gave us the virus that evolved into COVID-19, collaboration across borders was the way the world has been heading.

A golden clay age: The Brickyard (1990s)

Venues have effects on creativity of which we are barely conscious. One venue at O'Reilly is notable in this way: our office building on Sherman Street in Cambridge, Massachusetts. The next section describes some of the innovative thoughts that floated through this building. But the setting deserves some attention of its own.

The building was called the Brickyard, although someone claimed that it never held a brick-making factory and that the actual brickyard was around the corner on Bolton Street. No matter. Our Brickyard overflowed with old-time charm and quirkiness, like the mill at Maynard where Judy had worked after Digital Equipment Corporation took it over and launched a fad among high-tech companies for remodeling old New England manufacturing spaces.

Our occupancy of the Brickyard was quite recent when I came in 1992. Physically, the Brickyard was laid out around two angled staircases. This design was camouflaged by the walls dividing the companies that shared this old building. One staircase was plunked down in the center of our main lobby, while the other staircase was in the public lobby that led from the street to our entrance. The mirror-image staircases would amuse our children, who came to Christmas holiday parties and circled up and down between floors on the two staircases. Various other doors in the building led to other companies, so that people attempting to pay their cable bills would routinely wander into our office.

Why did the Boston-area office end up here after Tim O'Reilly moved to California? He rented the space from a computer consulting firm whose founder he knew, so the two companies initially cohabited the space and intermingled freely. Because our company grew by leaps and bounds while the consulting firm's clients dried up, we took over more and more space and finally had a large contiguous area all to ourselves.

Overall, the Brickyard space could be described as funky, with an atmosphere I compare somewhat to a college dormitory. Like university students, we alternated hard work with intimate moments. Years later, my middle-school-aged child Sonny (who later identified as non-binary), in order to raise money for their youth orchestra, brought their viola into the lobby under our staircase and played a cello suite by Bach. I invited everyone in our office and we enjoyed a good turn-out as well as some generous donations. After

a couple of movements, I gently suggested to Sonny that they had played enough. I had underestimated the appeal of Sonny's talent, worried just about boring my friends. The concert-goers protested vigorously that they wanted to hear the rest, so Sonny finished out all six movements.

We were all much closer than mere coworkers. I remember when one woman had a miscarriage, and we put together a scrapbook of personal poems, art pieces, and other artifacts to express our shared grief.

Our technology, apart from Barton Bruce's intriguing and intimidating racks of servers behind glass, was quite simple. Nobody had a personal computer, as I remember. Instead, each of us sat before a workstation connected by Ethernet to a single SunOS server with the name Ruby. Everyone had system administration privileges, so that if anything went wrong, the person who noticed it could fix the problem right away. All the staff were therefore immersed in the Unix and its X Window System environment (Unix with its corners rounded off, one might say) that constituted the scope of our product offerings.

Anyone who has occupied a historic building has been reading this section with anticipation of horror stories about the problems of working in the Brickyard. And yes, serious difficulties interfered with work from time to time. The building was plagued with all kinds of problems related to water. O'Reilly staff on the upper floor had to tune out the constant noise of workmen shuffling around the roof doing whatever they were trying to do toward the goal of reducing leaks. Bits and pieces of debris would land on desks and workstations. Once the workmen put a huge plastic tarp on one office ceiling to catch water that was dripping in, then forgot about the tarp for several weeks and came back to see it hanging precariously under a load of water. Water problems extended to bathrooms as well.

The technical staff also grumbled about the quality of the wiring available for electricity and our Ethernet cabling, although I did not learn the details and was not aware of any particular burdens the infrastructure placed on our daily work.

Schedules (late 1990s)

After many years, editing became second nature to me, and the challenges of performing my editorial role rested on logistics —just getting the book done. Not only could I quickly see what a text needed—sometimes penning a connective paragraph immediately, other times waiting a few hours in order to bring a more abstract perspective—but I learned how to explain my reasoning in such a way that the author would accept and feel grateful for my changes. I often ran through an initial high-level round

of commenting without touching the original text, but always asked for access to the source of the document for a round of more intensive editing.

I invested myself in every book, and it became an element of pride not to cancel a book. Of course, books would disappear from my roster for various reasons, usually because authors got too busy or lost interest as they discovered how hard writing turned out to be. My roster of active books usually hovered around 20, but only six or so actually got published each year; that was enough for me to make a profit for O'Reilly.

Once having trusted an author enough to champion their project and sign a contract, I was loath to tell them they could not succeed. I would simply present an honest assessment of their writing and an opinion about how to achieve our professional standards. If they realized it was beyond their abilities and gracefully withdrew, that was fine. Hardly ever did they persist in bad writing to the point where I or my managers would have to boot them out.

My fidelity to authors may seem excessive, but it has paid off repeatedly. Several times I watched authors blossom as we struggled through difficult passages. For some, English as a second language made their work hard to understand, but they mastered English as we persevered. I usually could tell when poor writing, whether through poor knowledge of English or some other lack of skill, hid real treasures.

The strain of the editorial job lay entirely in meeting schedules. I am proud of the books I brought into the world, but I rue the woeful hours spent with my manager, the manager of production, or both on obsessive scheduling.

This obsession wasn't arbitrary—publishing can be brutal. During the 1990s, schedules were paramount, and any deviation from a speculative set of deadlines that we negotiated half a year before publication was considered not only a monetary threat but a sort of moral lapse. Bookstores—whose sales were very important during most of the company's existence—required us to commit to release dates three months in advance. This was understandable because they, too, had to plan their revenues. Barnes and Noble at one point increased this lead date to six months. And missing this date by one month in either direction was highly disruptive. If we were a month late, some booksellers would cancel the order or take down advertisements. If we were a month early, no one would be ready to promote or sell the book when it actually did hit the shelves.

A six-month lead time is probably an easy target for the average publisher of novels or books on contemporary topics. We saw in an earlier chapter the leisurely schedule that Scholastic granted their editors. These generous production schedules could well

frustrate authors who watched the conditions they documented change and competing books emerge while their manuscripts sat idly in someone's inbox.

But technical book publishers such as O'Reilly had very tight production schedules of three to four months, which we were continually trying to shrink. To promise a book long before the author had finished writing the material was not merely difficult—it became an exercise in telepathy.

So I was repeatedly on the hook for some author who was going through a divorce or was suddenly gripped by the impulse to create a start-up company and started missing deadlines. I always implored authors not to do anything during their writing that would put their commitment to us at risk, but they would not put other parts of their careers on hold for a mere book.

What I learned during this time is an appreciation of book development as a time of mutual understanding and growth for both author and editor. A delicate psychology informs the editor's choice about how much advice to offer and how to make it feel empowering to the author. When the writing process falters, only sensitivity toward the author's circumstances, and a detailed memory of his patterns of interaction, can help make the decision about when to wait, when to cancel, and when to bring in help. Managers have sometimes been useful in tossing around possibilities and helping to determine responses to authors, but number-crunching offers no solution.

My manager at the time, Mike Hendrickson, once claimed I was passive-aggressive. I can't prove him wrong, because I have noticed times in my life when I acted passive-aggressively. Some may say that writing this memoir is a case in point. But truly, I do not believe my struggles during these years were passive-aggressive. There were real factors beyond my control that made it impossible to meet all the managers' expectations.

This explains my annual ratings over the years. People have told me repeatedly that I was O'Reilly's best editor. I think that's basically why I survived there so long. But during the many years when management employed a standard five-level rating used widely in many industries for annual reviews (where one "meets expectations", or "exceeds expectations", etc.) I never made it into the top level. No matter how many sales I was responsible for or how highly praised my books were, some lapse in scheduling or communication downgraded me in the end.

Now to recall: Frank Willison (mid 1990s)

In our Sherman Street days, the manager Frank Willison became an influential figure. He was our anchor that held the staff together during times of change, and was equally beloved for his sagacity as for his light take on life. Every communication from him contained humor and a profound tolerance.

Whenever a Sherman Street employee left the company, Willison would write a piece of doggerel and read it to an admiring crowd gathered from across the office. Some of those comic poems contained caustic observations that would not have been appropriate to share outside the office. But no one could ever be angry at Willison. He was too good-hearted, too well-grounded, too obviously concerned to do what was right. (The farewell parties disappeared during the following years, and CEO Laura Baldwin eventually told us that she felt it would not give a productive message to celebrate people who were leaving.)

Willison was not alone in performing health checks on his staff. The general ethos at the company was to protect its employees. We felt we were there for our personal self-realization as much as for the product. Such an ideal certainly couldn't be respected all the time, but I thought that lapses were rare and noteworthy.

Tim O'Reilly often expressed this ideal through the metaphor of a cross-country trip. You need to make sure to stop at gas stations so you can get to the exciting tourist destinations you've chosen to see. But the purpose of the trip is the tourist destinations, it's not a tour of gas stations. In this parable, the tourist destinations represent our job satisfaction, while the gas stations represent profit. Although Laura Baldwin exuded a "let's get down to business" demeanor that contrasted with Tim's cool swagger, I sensed that she also put first the health and growth potential of her employees, along with a conviction that the company must keep its mission in focus.

I took special note of Willison's serious side as my interest in my Jewish background grew and I engaged him in discussions about religion. He told me how reverently he considered the eucharist in his Episcopalian Church. He complained that after blessing the wine and handing out what was needed, the priest would pour the unused wine back into the battle. Willison disapproved of this thrift. He felt that if the church presented the wine as the blood of Christ, it should be treated more reverentially.

Willison unfortunately had health problems. I know that he drank too much caffeine. I remember a dinner we held, during one of our editorial retreats when we were all together from breakfast till lights out. As the waiter came around to offer coffee and desserts, the person next to me ordered decaf coffee. Coming next, Willison told the waiter, "I'll take the caffeine from his coffee and order a caffeinated coffee of my own."

His health problems, whatever they were, culminated in a heart attack in his Brickyard office in 2001. Someone tried CPR and a defibrillator, but nothing saved him. Willison's death was of course a grievous blow to his family. The office staff felt adrift as well. When the September 11 attacks on the World Trade Center and other targets followed a little later, we felt that the whole office was at the breaking point. O'Reilly management heard our cry, and scheduled a day-long retreat for all Sherman Street staff. We dealt there with our bereavement as well as with organizational culture and problems we were dealing with day to day.

Field trips for the mind and soul (early 1990s)

The variety of projects I describe in this memoir should give you an idea how many doors my career opened for me. I remember, many years ago, being with friends who worked in the computer field and hearing them compare how frequently they changed jobs. One said, "I never stay at any company for more than a few years," while another said, "I find myself changing jobs on the average of once every eighteen months." I just said, "My job is different every day."

Managers at O'Reilly routinely brought people through the Sherman Street office to broaden our outlooks with insights from other fields. One notable visitor was network expert Carl Malamud, a rollicking personality. He later gained fame by publishing records submitted by companies to the Securities and Exchange Commission, thus creating a new era of corporate monitoring. His courageous work might be considered the opening volley in the open government movement, which O'Reilly later championed under its Government 2.0 initiative.

Another surprise visitor was Mai Cramer, well known in Boston as the host of the popular public radio show Blues After Hours. Cramer came not in her role as radio host, but as a designer from Digital Equipment Corporation. I believe she was friends with our chief designer, Edie Freedman.

We also took our intellectual stimulation out of the office—for instance, on a visit to our printing firm. The boldest outings were a couple of wilderness trainings we held at the Habitat Wildlife Sanctuary, run by the Massachusetts Audubon society. Although the sanctuary is just a few square miles in size, it's enough to feel far removed from urban bustle. In fact, the sanctuary is just a half hour walk from my suburban home, although I didn't realize that until many years later during the COVID-19 shutdown, when I took long walks for my spiritual and physical health.

Our trainer regaled us with stories of Indian scouts and helped us appreciate the interconnectedness of our environment. We got a sense for how animals, ranging from frogs to bluejays to coyotes, navigated open space. Don't ask how such lessons make one better at producing computer books, the goal was personal enhancement that could lead to unanticipated contributions to future projects.

With these lessons, reinforced by the mindfulness one of my brothers taught me during hikes in Marin County, California, I decided to strengthen my sense of awareness by going back to Habitat on my own and wandering its paths, once every few months. I followed the teachings of our trainer, finding a quiet spot and staying for half an hour to see what was going on. I noticed that I was increasing my appreciation of the world's wonders in such simple things as how the layout of trees of different sizes reflected the ways they protected or inhibited one another.

One evening, I noticed the dark falling faster than I had expected (thick trees will produce darkness even when sunlight is shining outside them). I managed to get back to my car before getting lost, but the experience informed a spooky forest scene in my parody of Nathaniel Hawthorne, "The Bug in the Seven Modules".

I didn't learn the lesson I needed from this close call, and brought calamity on myself a few months later. I visited Habitat again mid-winter and slipped on the ice coming back. My right leg couldn't hold my weight, so I crawled the last quarter-mile or so through the snow. Luckily, some other visitors were near the entrance, because I clearly couldn't have driven away. The visitors got me an ambulance, and the hospital told me that I had broken my ankle.

I was lucky that day. What if I had fallen in similar fashion at night-time, with no one around?

Probably, the ankle that broke had been weakened by an earlier auto accident, just a couple years earlier, a driver had knocked me over as a pedestrian in the dark near my home. Because the auto accident broke my wrist along with my ankle, Judy suffered substantially taking care of me and the whole household, while I spent many weeks trying to train Dragon Dictate to recognize speech. The speech-recognition software is now distributed by Nuance, and has probably seen worlds of improvement. The second break to my ankle did not thrill Judy any more than the first, although the recovery was faster. I decided not to risk putting us in such a situation again, and therefore relinquished my solitary nature walks. Instead, I decided I would expand my world intellectually from my easy chair. I stepped up my poetry writing about that time.

Returning to the topic of my joining O'Reilly & Associates as a full-time employee, there's a discrepancy between the official record and my own memory of my first day. The official record places my hiring as Monday, November 9, 1992, whereas I remember it being exactly one week earlier, on November 2. No one but an astrologist at this advanced time would care about the discrepancy, and perhaps even the astrologist would consider it an acceptable margin of error.

What I do remember clearly about that day was the sense of wonder mixed with excitement that I felt upon entering an office that I shared with a long-time employee, Lenny Muellner. In truth, Professor Leonard C. Muellner had earned tenure in the department of classical studies at Brandeis, but supplemented that career with a part-time job doing various technical tasks in our Brickyard office. This information came out during our initial repartee and led later to numerous interesting conversations, such as his cautiously positive response several years later to a news item claiming the discovery of the long-lost grave of the emperor Alexander the Great. By the time the tools team was eviscerated, Muellner had left O'Reilly to take a new job revitalizing the Center for Hellenic Studies at Harvard—a role that eventually brought him around again to digital publishing.

Equally strong in my memory of entering my office was a large cardboard box under my soon-to-be-occupied desk, containing documentation for a company called Cygnus. This was one of the most important companies that entered the early free software movement. Their role was to package and support the compiler tools from the Free Software Foundation. Later they created an environment called Cygwin to run these compiler tools and other Unix-style utilities on Windows systems, a valuable link between those two very different and hostile worlds. But I had no idea why the volumes of documentation should be under my desk. I asked Muellner, "Is every day here going to be as surprising as this?" He laughed in response. Further inquiries around the office let me know that Cygnus was seeking a partnership of some type. I suppose that even on my first day, I was known as a proponent and intimate of free software.

First editing project and a lurch into our future (1992)

I wrote and edited my first books for O'Reilly as a freelancer. Following my first project, which was to update O'Reilly's book on the little programming utility called Make, I proposed that the company write a book about a major update to the Fortran language. I had devoted a lot of time to this language at two scientific computing firms I had worked at earlier. Engineers were reluctant to give it up for other languages offering modern conveniences, and the compiler writers tried to give Fortran a new lease on life by incorporating some modern syntax and constructs from other languages. The project was thus called Fortran 8x until it

became obvious (as one leader joked) that the "x" would have to be hexadecimal. In the end, they missed the 8x deadline by just one year, and Fortran 90 became a standard.

The company received a proposal from an unknown author named James Kerrigan who wanted to write a Fortran 90 book, and I argued for its approval so assiduously that Tim O'Reilly offered me the job of editing it. I don't know how I managed to fit in the work while holding down my full-time job and raising two small children, but I had a wonderful time and discovered for the first time the pleasures of building a personal relationship with an author.

Our book came out around the same time as a few dozen other books on the topic by other computer book publishers, all of whom shared with the Fortran compiler writers the optimistic assumption that scientists and engineers would honor many decades of seniority and remain loyal to the language. But none of the books sold at any rate worth the cost of their development.

Still, I had reasons to be proud. There were indications that our book topped the pack, most notably an offer by Microsoft to package an electronic version of our book with every copy they sold of their Fortran compiler (somewhat of an unusual distribution mechanism for a book back then). Our marketing manager, Linda Walsh—in fact, I believe she simply was the totality of O'Reilly marketing in the early 1990s—pursued the Microsoft offer eagerly at first, but ran up against their insistence that they get the book for free. Microsoft arrogantly claimed that merely bundling the book with their product would be such a wonderful endorsement that we should ask for no compensation on top of that. Linda may or may not have known the digital direction that the publishing industry would take later in the decade, but she knew a big swindle when she saw one. Microsoft never got our book.

Walsh was a super connector, an important trait in marketing. She and I were walking between Porter and Harvard Squares in Cambridge one day when we ran into Peter Salus, another super connector in the Unix world. Accompanying Salus at that moment was Rob Pike, one of the inventors of Unix. I wondered at that moment how many famous people I walked by unknowingly every day.

In these first years of my work at O'Reilly, I tend to talk about individuals rather than departments. We were a small company. A department was often an individual, plain and simple. Our international program consisted of a single person, Peter Mui. He did everything that later, at our height, we had six offices working on.

We had a print coordinator in those days, named Sue Willling. Keeping an eye on your printing company used to be a significant job. Although I certainly am no printing expert, I remember several times when things would go wrong. For instance,

once we sent a book to a printer who had a degraded version of a font we were using. The printer's version of the font could not distinguish between an apostrophe and a backtick (a rarely used character). All backticks were converted to apostrophes in the book. Because the book dealt heavily in Unix shell commands, backticks were scattered liberally throughout the book and were most definitely used differently from apostrophes. The book looked fine at O'Reilly, but was completely mangled by the printing company. The author hit the roof, justifiably, and the first printing had to be pulped.

Willing helped us avoid such embarrassments. But increasing standards made it easier to get books printed correctly, and our move to the Internet weakened the importance of a print coordinator. No such position has existed at O'Reilly for many years.

Alter ego: Praxagora (1992)

Coming to O'Reilly full-time was a gradual process for me, unfolding over some two years. O'Reilly had always been at the back of my mind because the O'Reilly employee Adrian Nye had written some of the X Window System book series as at the request of my earlier employer, MASSCOMP. The system administrator who set up the MASSCOMP workstation that I would use (not for the high-speed data acquisition the systems were designed for, but for routine testing and book development) told me that my system had copies of the X Window System books Adrian had edited and that Adrian was expected to come back to update them. Not realizing that Adrian referred to a man, I spent my first year or so expecting a woman to walk into my cubicle, say "Hi, I'm Adrienne," and take over my workstation.

My first glimpse of the new space O'Reilly leased at Sherman Street came a few months before I was hired. Tim O'Reilly seemed quite proud of the office, which was a big step up in size and professionalization over his previous office (not to omit the barn), and he wanted to show me around before interviewing me over lunch. This was not a formal interview, but more like a free exchange of ideas. We went out with a couple other people, including Mike Loukides, who had recently been hired as an editor.

Tim chose the nearby Joyce Chen's restaurant. I thought how there were a few other Chinese restaurants that were much better. I believe I was rude enough to suggest that to the head of a company who was about to hire me, and who was paying for lunch to boot. I don't remember the topics of conversation, probably because the computer field has changed so much and our understanding of it has evolved so richly since then that it would be almost as hard to re-enter the mind of a young technologist in 1992 as to relive the experience of a two-year-old child.

Tim had no need to interview me, having employed me on a freelance basis for some time already. My luck in landing the job at his company was a textbook illustration of the old boy's network. About a year and a half after I arrived at MASSCOMP, the technical writers' management job fell on Steve Talbott. Talbott knew Tim O'Reilly well, and he certainly knew Adrian Nye's work at MASSCOMP.

I wrote up some of my adventures at MASSCOMP, and especially my relationship of mutual trust with Talbott, in a chapter of the O'Reilly anthology *Beautiful Teams* many years later. Talbott was impressed by my mastery of tools, particularly the odd Unix utility called Make that was used to automate builds in an age before integrated development environments. It was coincidental that I designed my own build tools around this utility and that Talbott had written a 70-page book on it called *Managing projects with make* for publication by O'Reilly. He could not help but notice my creativity with both technology and writing. So he asked me to write a new edition of his book for O'Reilly. Not realizing what an honor that was, or for that matter how much work it would demand, I launched in and successfully finished the project at the perfect time: the week between the end of my employment at MASSCOMP and the start of my next job.

The book on Make conformed to two themes that have run through this memoir: the shock of unexpected tasks and finding a creative response. The shock was to discover that many different versions of Make were out in the field. It was only after finishing my draft and sending it out to technical review that I learned from my reviewers that the job was much bigger than I expected—that in fact it involved a whole new dimension of explaining differences between versions of the utility. Not only would Make behave differently on different computer systems, but many users wouldn't know which version they had.

How could I offer instructions to readers when the program's behavior was unpredictable? I took an eminently practical and creative solution: I provided a script that readers could run to determine which features were supported on their system. This script turned out popular in its own right.

At the end of the 1980s decade, MASSCOMP faded into irrelevance due to changes in the computer industry. It became either beneficiary or victim of an acquisition and gradually lost all its talented employees. Talbott made his way out before me, and after taking a job as an editor at O'Reilly called me and urged me to join him. But I felt more empowered working in the computer industry directly, rather than at a remove in a publishing company. Talbott explained that I would rub shoulders with the very leaders of computing by joining O'Reilly, but I rejected the advice at first and spent a year and a half at another company, the U.S. computer subsidiary of Hitachi mentioned in other stories.

I do not regret that detour, because I took responsibility for much tool development, got comfortable reading the source code for the Unix operating system, and learned to navigate Japan social mores. But problems both at Hitachi and the environment in which Hitachi operated soured the situation, and I gradually pulled closer to O'Reilly & Associates.

Meanwhile, I started to seek (years before the availability of web sites for personal self-expression) a way to air my opinions about the important interactions between technology and society. After joining O'Reilly I reserved the praxagora.com domain name, hoping to offer a web site that would offer a broad forum for opinions and insights about computer technology. For Talbott, I hosted a good deal of online essays about his views on computers, which ironically were quite negative.

Talbott was a competent computer technologist who earned a living creating value in computing but consistently warned about its risks to human consciousness and society. There is nothing new about such contradictions. J. Robert Oppenheimer ran the Manhattan Project in order to provide the free world with a defense against Hitler, but then spent the rest of his life trying to rescue the world from the atom bomb that resulted. Computer Professionals for Social Responsibility, where I spent the bulk of my volunteer efforts for 15 years, warned constantly about the downsides of the technologies in which we were expert, as did the Association for Computing Machinery's policy group. But Talbott was more of a Cassandra than the rest of us.

It turned out that no one else asked me for space on the web site associated with praxagora.com, and Talbott eventually moved his content to a different host. This was the right thing to do, because my view of computing and Internet access was largely laudatory while his was cautionary. But for twenty years, I kept a pointer on my web site to Talbott's the new site.

Wrapping up (1992)

MASSCOMP was apparently unaware that O'Reilly & Associates had absconded with the books Adrian Nye had written on MASSCOMP's dime and had thrown aside contract writing to develop the X Window Series into a massive money-maker. This story provides a tale of free content and commercialization.

⋙ The historic importance of the X Window System was its universality. The Unix operating system had developed for several years with no interface except a text-based green screen experience. Users learned tools such as Make at the command line and used editors such as my own beloved Emacs in full-screen mode. There were few graphical systems and no standard for graphics until a team at MIT created the X Window System. Their invention became the

> graphical interface for all Unix systems and still plays a role in the version of Unix offered by Apple for its Macintosh systems. Of course, it was free software, released under the MIT License with the simple mandate to go forth and multiply.
>
> Suddenly a raft of tools beginning with the letter "x" became part of the Unix user's toolset, and with this came an urgent need for documentation about how to create more x's for the Unix world to use. MASSCOMP had hired Nye to beef up the MIT documentation and make it more professional, as well as to write a series of programming guides that Nye produced with great aplomb. There is no doubt that he was a master of the difficult art I was also trying to practice, that of readable and actionable documentation for computer professionals.

O'Reilly's coup came from permissive licensing. On the MIT side, they let people do whatever they wanted with their documentation and did not require (as the Free Software Foundation does in their GNU series of licenses) that all changes to distributed versions of the code or books be contributed back to the original creators. O'Reilly could therefore intertwine MIT's massive contributions with the improvements made by Nye and others and put the O'Reilly copyright on it.

Furthermore, MASSCOMP management made no thought as to the value of documentation outside their own company. They imposed no contractual requirement on O'Reilly to ask permission or pay any licensing fees when O'Reilly commercialized the X Window Series. Had MASSCOMP requested a modest licensing fee in exchange for their funding of the project, their financial prospects would have been much better.

The X Window Series became transformational for O'Reilly. The company could now devote itself purely to publishing and quickly became a household name in the Unix field. Companies such as Sun and Hewlett Packard, expecting their workstation users to create X programs as a matter of course, bought up crates of O'Reilly X books and provided a full set (the series ultimately reached about nine volumes) with every purchase of a workstation. When the common X core splintered a bit and generated a variety of extensions, O'Reilly could afford to rewrite some of the series in different versions.

»»

> This exploitation of intellectual property has a curious parallel with another leading company in the tech world: Microsoft. Bill Gates started his company to provide IBM with an operating system for its new personal computer. Numerous companies had witnessed the popularity of early kits and cobbled-together personal systems. All these companies were pushing their offerings on the market in the early 1980s. IBM, an improbable contender because it

was a mainframe company, wiped out the competition not through any technical superiority, according to many analysts, but simply through the faith that major corporations continued to place on these three familiar letters.

IBM produced software as well as hardware, but perhaps lacked confidence in their ability to provide an operating system for this tiny device. They somehow discovered Gates, and he licensed software from somewhere else and provided it under the name MS-DOS to IBM. Again misunderstanding the value of software—which they were used to distributing for free—IBM left total control over MS-DOS to Gates. And as knock-off PCs emerged from other companies, MS-DOS became the default standard for everybody. Although IBM throughout the 1960s and 1970s was a feared monopoly and a target of U.S. government anti-trust action, the age of the PC saw the end of an IBM monopoly and a shift toward the dominance of the two features all PCs had in common: Intel chips and MS-DOS, followed by Microsoft Windows.

O'Reilly is quite different from Microsoft. The publisher had established a stronghold in computing with a highly respected line of Unix books like the one I edited, *Managing projects with make*, before they started the X Window System series, and they never attained the impregnability in their industry that Microsoft enjoyed for decades. Both stories, however, show the power of control over key resources and how a shift in value can turn long-disregarded side products into indispensable commodities.

Thus it was that I joined Nye at O'Reilly instead of Nye joining me at MASSCOMP. Nye played an important editorial role until he decided that his life's passions took him out of publishing. He adopted early on a concern for climate change, which we all had heard of but was not in the 1990s the clear danger to life that we later recognized. On the other hand, Nye was learning how to fly a plane and loved that hobby. After he left, one of my fellow employees mocked Nye's professed concern for the climate, dubbing him "Private Plane Nye". But I have never seen a contradiction between adopting one or two carbon-producing elements of lifestyle and caring about the Earth. All of us contribute more carbon than we should to the atmosphere, and we know that the solution cannot come from individual sacrifices but from a collective change to the whole structure of the economy and society.

For Nye to sacrifice his plane, or for any one of us to sacrifice a long commute to a job where we do good or the loving gesture of giving someone flowers would mean to relinquish our individual creativity. To emerge from climate disaster, particularly at this seemingly impossible moment, requires extraordinary and probably unpredictable accomplishments from the human race. The crisis calls on all the innovativeness, all the passion, all the personal investment of heart and mind that we can muster. That joy in creation

was what Tim O'Reilly and his company celebrated from the beginning, what kept me there for nearly three decades, and what gave me the platform and the unlimited mandate to contribute to so many causes.

Andy Oram
October 2020

Author's home page

If the computing history in this memoir interests you, check out some other articles where I cover specific historical topics:

From Unix to Linux: Key trends in the evolution of operating systems (November 13, 2020)
How the RESTful API became a gateway to the world (April 14, 2021)
The many meanings of Linux (September 17, 2021)
Open, simple, generative: Why the Web is the dominant Internet application (August 17, 2021)
Who's building businesses around free and open source software? (March 16, 2021)
How the Apache project boosted the free and open source software movements (March 24, 2021)
How free software contributes to cloud services—and what the cloud vendors give back (September 15, 2020)
Ten factors behind the popularity of microservices (July 8, 2021)
How the X Window System influenced modern computing (June 16, 2021)
Awk: The power and promise of a 40-year-old language (May 19, 2021)
How hashing and cryptography made the internet possible (September 20, 2022)
The network neutrality debate: It all depends on what you fear (August 30, 2010)
Fenced-off culture, the privatized Internet, and why book publishers lean on a 30-year-old doctrine (September 29, 2022)
Network neutrality: Distinctions and controversies (September 12, 2010)