Features | Management | News | Rational Reader | Technical | Franklin's Kite | Rational Developer Network

Subscribe
Contact Us
Submit an Article
rational.com
Issue Contents
Archives
Mission
Staff

Driving Technology: Alpha Geeks, Open Source, and More

An Interview with Tim O'Reilly, President of O'Reilly & Associates

Download pdf version (132 K)

Illustration

Tim O'Reilly *is founder and president of O'Reilly & Associates, Inc., the 300-employee publishing powerhouse based in Sebastopol, California. The company has been publishing high-end technical manuals since 1984 -- no "dummies" need apply -- and more recently began sponsoring conferences and providing electronic subscriptions to books via its Safari service. The company's online division, the O'Reilly Network, manages a technology information portal at oreillynet.com as well as ancillary sites for key technologies, such as xml.com, onjava.com, and perl.com. Another key project has been CollabNet, established with Apache Group cofounder Brian Behlendorf, which evangelizes open-source development methodologies within Corporate America. With all of these offerings, O'Reilly thinks of his company as an information provider, and not only a book publisher.*

In a recent informal talk at the Softpro bookstore in Burlington, Massachusetts, O'Reilly took developers and other attendees on what he called a "random walk through the contents of my brain drawers." What follows is based on both this presentation and follow-up conversations with Rational Edge *writer* ***Johanna Ambrosio***.

Johanna Ambrosio for *The Rational Edge*: What changes do you see on the technology horizon right now? Will developers have a role in driving these changes?

Tim O'Reilly: Developers are already driving them. As science fiction author William Gibson once said, "The future is here; it's just not evenly distributed yet." I think this has always been true for technology changes, because they're led by "alpha geeks" who, when they can't find software to do what they want, go ahead and create it themselves. They feel very comfortable pushing the envelope in whatever way makes sense to them. And they're the ones who help make a new technology easier for other people and broaden its use.

Technologies like Perl and the Internet, for example, started with the alpha geeks. Typically, they struggle through a new technology and use "it" before the technology even has a recognizable name.

Web Services will be a big thing, I know, because alpha geeks have been at work on them for years. They're already happening, just not in a standard way yet. Really, Web Services is just a fancy name for program-to-program communication over the Web, typically with a minimum of human interaction. Web spidering -- using a custom program to extract data from a Web page without going through a browser -- is a precursor to Web Services. Programmers just figure out the calling parameters of a CGI (Common Gateway Interface) script and send them programmatically, treating the URL as a kind of command line.

For that matter, when you go to a Web page and see an ad banner, for example, you're invoking a Web Service -- when you load the page, your browser also makes a call to DoubleClick or AdForce, via a hyperlink that dynamically inserts the ad. Maybe that's a service you don't want. But it's a cooperative process using lightweight techniques. The point is that programmers have been figuring out how to overload the simple client-server model of the Web for years. Now, we're just trying to standardize that "data Web" with XML-based protocols like XML-RPC (Remote Procedure Call) and SOAP (Simple Object Access Protocol), and people are giving a name to the old model of URL as command line. And now we're starting to hear about an alternative model Representational State Transfer (REST).[1] But the story's not over yet -- there will be more pieces that click.

JA: What do you see happening with the Internet over the next several years?

TO: I think the real Internet era is just starting. I see the Net becoming *the* environment in which we live and work -- the center as opposed to the edge. This environment will include more Net-connected devices that are sometimes on and sometimes off, and the ability to take your data with you in your pocket. The PC is still the dominant paradigm -- it's where the "big money" still is. It's just like in the 1980s, when the big money was still in mainframes, even though the PC had already been introduced.

The big challenge will be what the Internet operating system will look like. It won't look like the current generation of either .NET or SunONE or anything else that's out there right now. What we need is to get to the next step from today's situation, where there are a bunch of non-standardized techniques that only the alpha geeks know about and can use. We're in the roll-your-own phase of Internet development. Now we need someone to package up all the really useful bits -- to put all the great peer-to-peer and other tools together as part of a "standard" platform that *all* developers can use to create software. It's like when Microsoft came out with the Win32 API; they told developers that, instead of having to worry about the thousands of drivers for the PC, they could just write for the APIs Microsoft provided. Someone will need to do that for the Internet platform. Wouldn't it be great if someone could put MapQuest's functions into an operating system, for example? That way, I could put a query to find the distance between any two points into any application I wanted. You need to expose these things to the programmers, not just the users. Give us some interfaces!

JA: So you're saying that platform-based standardization -- what we already have on the PC -- is what lies ahead for the Internet, too?

TO: What I really think is that everyone's trying to jump on the Internet bandwagon and to reproduce the Microsoft platform lock-in. Netscape thought they could do it through the browser. Sun hoped they could do it with Java as a new universal virtual machine. (And they may yet succeed.) Microsoft is now trying to do it with .NET.

But I also think that the industry has gotten a bit ahead of itself, in that it's hard to sell a new "operating system" whole cloth. Instead, what has to happen is for a half-baked OS to emerge, with lots of problems that nonetheless highlight the issues. Only then can someone solve the various problems with a systematic solution.

Consider Web Services, for example. There's a lot of potential in both J2EE and .NET, as well as in XML standards like SOAP, but what is missing are the actual programmable components. These are the equivalent of all those PC devices that were so burdensome to write drivers for, and for which Microsoft offered a solution with the

Why Did O'Reilly Go Ape Over Animals?

Technical professionals the world over instantly recognize O'Reilly books by the intricately rendered illustrations of animals on the covers. Tim O'Reilly talks about why he decided to use animals to illustrate technical manuals, and addresses other publishing-related questions.

Q: Where did the cover animals come from?

A: It was one of those things -- an open-source thing, in an odd way. Back in our early publishing days, we had seven books that we called "Nutshell" handbooks. They all had the same design, with brown paper covers, and so on. We realized that customers weren't distinguishing one book from another because they all looked alike. One of our writers talked it over with her housemate, a graphic designer named Edie Freedman, who thought that a bunch of the Unix commands, like sed and awk, sounded like animals. She drew up some

Win32 API.

To me, these programmable components are all the various large Web-facing databases, and the equivalent of the build-your-own-driver school of programming are the Web spiders that access those services programmatically. Web spiders, including unauthorized interfaces built by screen scraping, are one trail of breadcrumbs we need to follow when looking at the functionality that an Internet operating system will need to provide. Distributed applications like SETI@home[2] and Napster are another.

The bottom line, though, is that whatever the Internet operating system will become is still emerging. Like most technology breakthroughs, we'll figure it out after it's already happened.

JA: What role will open source play in this Internet era?

TO: A major one. People think open source is about licenses -- but it's more significant than that. In fact, a lot of the value of open source has been obscured by all the debate about licensing. Open architectures allow people you don't know to invent things you haven't thought of that work with whatever you already have. If you decide on a few simple rules, things will work together, and you don't have to build a lot of special interfaces.

Here's what I think open source promises:

covers over the weekend, and we loved them. She gave them to us for free, too. Of course we later hired her, and she's still our chief graphic designer.

Although authors don't tyically get to pick their own animals, the designers will respond to interesting, offbeat suggestions. For example, Larry Wall picked the animal for *Programming Perl*. He said he wanted a camel because it's like Perl -- it's ugly and can go long distances without water. Most of the time our graphic designers select the animals, and there's always a connection between the animal and the book topic. It's not obvious -- it should make you think, create a spark. Once we had a book about Python with a python on the cover, and I wasn't too happy about that; it should have been a rat or maybe a peccary.

Q: Are the authors of your books the alpha geeks you referred to earlier, or others?

A: We definitely watch the alpha geeks! We try to get them as our authors, or pair them up with writers who can capture what they know. In fact, we characterize our core business as "capturing the knowledge of innovators," not just publishing.

Q: How do you keep up with bleeding-edge technology if it takes months to produce a book?

A: It's not as hard as it looks. First off, we have an online portal, the O'Reilly Network (www.oreillynet.com), that allows us to cover fast-breaking technology. Similarly, we work a lot of bleeding edge stuff into our conferences (http://conferences.oreilly.com), especially the upcoming Emerging Technologies conference.

We only do books when it looks like a technology is around to stay for a while. And it usually takes a while for technologies to "cross the chasm," so this sequence of businesses really lets us work with the users as the technology propagates out to wider and wider groups.

- **A system architecture that makes it easy for independent contributions to be accepted as integral parts of the system**. A loosely coupled, protocol-centric architecture tends to support collaborative development, as does the use of independent modules rather than a single massive code tree.
- **Reasonably ready access to source code**. This need not be on a

"free for all" basis. In fact, many corporations have more internal developers than many successful open source projects have total developers. And in many other cases there is shared source between companies and their customers. This has led to the concept that has somewhat waggishly been called a "gated open source community." Meaning that it's not totally open, in the purest sense, but it's not totally closed, either.

- **A communications network that allows independent developers to contact each other and work together over a wide area**. One of the most striking things about most open source projects is how they aggregate talent and interest over wide geographic areas, rather than requiring people to belong to a single institution or local area. Linus Torvalds (a Finn who now lives in California) has -- as his #2 man on the Linux kernel -- Alan Cox, who lives in Wales. The Perl core team has people in California, Australia, the UK, and Germany. The Apache core team also has people from a half-dozen countries. It's the Net, not just the open source license, that makes these projects possible.
- **Tools that make it easy to get the latest source, and to build and modify it**. In the low-bandwidth days of usenet, patch was perhaps the single most important tool after usenet, e-mail, and ftp. But nowadays, we're talking public RCS (Revision Control System) access over the Web and Web-archived mailing lists for that same purpose.
- **A cooperative attitude rather than a proprietary one**. This need not extend all the way to an open source or free software license, but there definitely has to be a will to cooperate.

JA: Some people see a conflict between the open source approach you've just described and the "closed," very structured development that goes on in most corporations. Do you think those two things can live together happily?

TO: Absolutely. Open source is really a distributed development paradigm. It's a way of developing software, a side effect of network computing. It's wide-area software development, like what we're doing at CollabNet, which I'm very proud of. We're evangelizing to get companies to use the open source approach I described above, even if they don't use open source code itself. If what you're doing as a company is too closed, then you're not going to get people to invent on your platform -- you'll miss out on those alpha geeks we were talking about.

Once you make the decision to go for an open source approach, then the question is: How do you get consensus with a widely distributed group? If the Apache group can build software with people who are spread out all over the world, then so can others. There's no conflict there: Good development is good development. In most big companies, you have multiple types of projects going on anyway -- there's a freewheeling, let's-see-what-we-can-do side and a more structured side. No one methodology is going to work for everything all the time, so open source will just be part of the mix, one of the choices.

JA: You mentioned Apache. What's made it work? And what other open source projects do you consider to be successful?

TO: Apache is probably the most interesting project, precisely because it doesn't have a single visionary leader. It was started by a group of users whose "vendor" (NCSA) abandoned their project; all the developers went off to start Netscape.

They started with source code, a mailing list, and a desire to share their patches. They built a set of techniques for wide-area collaboration and voted on patches. They also kept to a vision of modular code rather than allowing feature creep. They kept the core Apache program small, while creating add-on modules for Java support, XML, etc. as separate projects.

I also like to point out that the original Unix development at ATT Bell Laboratories (later joined by the University of California at Berkeley and folks at other universities) was by all measures except licensing an open source community. The license wouldn't meet today's standards for free software or open source, but the community showed all the characteristics celebrated by Eric Raymond in *The Cathedral and the Bazaar*. The key to open source, as described in this book and accomplished so successfully at AT&T, was the ability and desire of programmers to freely share their code. Raymond's whole point is that Internet-enabled collaboration and free information sharing, not monopolistic control, is the key to innovation and product quality, and I agree with that.

The key example in my own experience was with AOL and MSN versus the Web. Both AOL and Microsoft had ambitious plans for online content, and as a content provider, I was in discussion with both of them.

Behind door #1: Sign our contract, use our proprietary tools, get on board our marketing machine, and we'll make you rich and famous, but only if you pass our hurdles for being interesting and important, and follow our rules.

Behind door #2: Download this free software, use this simple syntax (which you can learn by copying from what someone else has done using "View Source"), and put up whatever content you want. No one is asking who you are or whether you qualify.

Which of those models led to the development of the rich online content marketplace we now all take for granted?

JA: Do object-oriented techniques and technologies figure into open source?

TO: In theory, OO and open source have similar objectives: code reuse. And it's certainly true that my idea of loosely coupled architectures fits somewhat with the OO dream. But at the end of the day, the architectural level that I'm talking about is independent of language or programming technique. It has to do with fundamental system architecture, not with the architecture of a particular program. For example, Unix and the Internet have a loosely coupled architecture. Windows does not.

JA: What about the bottom line? Will open source have a positive economic effect on the technology industry?

TO: I think it might revive at least one trend that was strengthening just before the dotcom bust: Open source elevates the work of individual programmers and gives them status, independent of their organizations. Project owners take that with them when they move to another company, and that shifts the balance of power between companies and their engineers -- much like the free agent system changed the balance of power between athletes and their teams.

Also, open source can certainly commoditize some previously lucrative markets. Bob Young of Red Hat once stated that his goal was to shrink the size of the operating system market. And it seems pretty clear to me that the support for Linux by hardware manufacturers like IBM and HP is driven both by potential cost reduction for their own proprietary system development and by a competitive advantage against market leader Sun. Similarly, there may be support for Linux driven by avoidance of Microsoft licensing fees. There are often complex dynamics in the business ecology behind these economic choices.

But this is a complex issue, and it's easy to overstate the role of open source. For example, no one is making money by selling Web servers or Web browsers. But note that even though both are free of cost, the dominant browser is proprietary, and the dominant server is open source. What that says to me is that there were other drivers than open source for the commoditization of Web software.

The overall economic benefits can be very positive, though. Vendors recognize that open source allows companies and users to cooperate more effectively, and this can increase user satisfaction and decrease vendor costs. Not every feature can be developed commercially by a vendor, but if users can extend the product themselves (whether through open source or an extension mechanism), then they can increase their satisfaction with the product at little or no cost to the vendor. And if they can share those extensions with other users, they may grow the total community.

In the early days of my company, for example, we did a series of books on the Pick operating system as well as on Unix. Pick vendors tried, for the most part, to keep their boxes closed and their users in the dark (in order to maximize vendor revenue opportunities), while Unix vendors empowered their users. The Unix philosophy led to a robust aftermarket -- not just open source software, but also books like mine that fed the whole system and grew the market for everyone. Not so for Pick.

I also think there's a world of economic opportunity in software services -- and by services I mean "We make our money by using open source software to deliver a service." It was Rick Adams of Uunet, the author of Bnews and SLIP, and not the founders of Red Hat, who figured out how best to commercialize free software. Rick didn't put usenet news and e-mail in a box and try to sell it. He offered a reliable, high-performance subscription service -- and gave birth to a whole new type of company, the commercial Internet service provider. And he continued to give back to the free software world through his support of the Internet Software Consortium, the nonprofit that funded ongoing development of both netnews and BIND, the software behind the domain name system. Rick doesn't give a fig for free software or open source licensing or rhetoric, but he figured out how to get his software adopted via free source distribution and commercialized through subscription-based usage.[3]

JA: What do you predict will really matter to the industry when you look back in four or five years?

TO: To tell you the truth, what I really want, what I care about, is that this industry continues to be as much fun as it's been for the past twenty years. A lot of industries get boring as they mature. Alan Kay, who's still at Xerox PARC, said the best way to predict the future is to invent it. It's much more interesting to go out and build the future than it is to talk about it. I look forward to seeing what people build -- someone in this audience may create something just wonderfully compelling.

Notes

1 REST, a term coined by primary architect Roy Fielding, refers to a networked software architecture. More information can be found in Fielding's dissertation at: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

2 SETI@home is a project that involves hundreds of thousands of individuals donating unused cycles on their Internet-connected PCs to help search for extraterrestrial intelligence. The program is sponsored by the University of California at Berkeley.

3 To read more about O'Reilly's views on what's next for open source, check out his October 2001 article in Linux World magazine: http://www.linux-mag.com/2001-10/trench_01.html

tre icon

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!