CHAPTER ONE

Tim O'Reilly on Leadership

Tim O'Reilly started the company that became O'Reilly Media which published this book (and our others, as well). We've always personally admired him and the company that he founded, not just for his impact on the publishing industry but also for his impact on the larger world of software and software development. We've worked with people from almost every part of O'Reilly for years, and we recognize a great team when we see it. We wanted to talk to Tim to hear his ideas about how he built his team and continues to bring out the best in them.

Jenny: One thing we noticed when we started putting together this book was that everyone seems to have a slightly different definition of "team." And it turned out that some of the stories we included talk about that question explicitly: whether a team is

always people who are together temporarily to build some specific project, or whether a team can be a group of people who've never met.

Andrew: We kind of pulled a fast one and just left it open to interpretation. So how do you define a team?

Tim: My own experiences are around running a company. Yes, there are team experiences in that. But most of the reflections I have are around the broader question of how you exert leadership. Let me start at the top, with a few thoughts about leadership and management, which are part of the whole team thing. I'm not quite sure where the boundaries are.

There are two quotes I want to start you off with. One is from Harold Geneen, who was the guy who started ITT, which was really the first modern conglomerate. And he said, "The skill of management is achieving your objectives through the efforts of others." So that's kind of an interesting perspective. And the question is of the different styles of doing that, where some of them are very directive. This is the classic "manager"—the idea of somebody who figures out what needs to be done, and who needs to do it, and builds the teams with the roles, and so on.

While I completely subscribe to the concept, because the skill of management is indeed achieving your objectives through the effort of others, I have always worked with the framing of another quote, which is actually about writing. It's from Edwin Scholssberg, who wrote a magazine article I read early in my career, and it's probably one of the seminal things that took root in my brain. He said, "The skill of writing is to create a context in which other people can think."

Jenny: So can you think of how you apply that idea—"creating a context in which other people can think"—to software teams?

Tim: I'll make an observation here, and it relates to something I call the architecture of participation. In 1998, we did a book called *Open Sources*, and we did interviews for some of the people who produced their essays on open source. And I don't think it made it into the final book, but there's something that Linus Torvalds said in an interview that stuck with me: "I couldn't have done what I did with Linux for Windows, even if I'd had the Windows source code. It just wasn't architected that way." And it really set off a chain of thinking in my mind about the architecture of open source projects, and how they're designed to allow that sort of free-form improvisation. Because there are rules that are laid down.

Andrew: It's funny to hear Linus talk about not being able to do something—it's a good reminder that SourceForge.net *is littered with open source projects that never went anywhere.*

Tim: I think one of the reasons why certain projects fail is because they're mixing and matching from the wrong systems. We have to have a system that has a fundamental characteristic that there are small pieces that people *can* work on independently. I think this is why, for example, people have said that they'll do books as Wikis, and it hasn't

really taken off. Why not? Because a book is a fairly large, complex thing with a single narrative thread. Wikipedia is a set of pages, and the atomic unit of content is something that a single individual can make a plausible promise at, and other people can update and tweak. And the whole is the sum of many, many such small parts. I think, for example, that there are certain types of works that lend themselves to that kind of collaborative activity—being more free-form precisely because they're designed in such a way that the pieces fit together.

Unix is designed a little more like a set of LEGOs, where the design principles are that you have these "innies" and "outies" and they snap together. Thinking about pipes and filters and all of those kinds of things—that people can write completely independent utilities with the knowledge that they just fit. Most programs read standard in and wrote standard out, and there were a few simple rules.

Similarly, every atomic piece has a manpage. I've always thought, if I were given a choice between Windows with open source code under, say, the GPL, versus Windows with a manpage for every little bit of code, what you could swap out and what you could change. Yes, you'd have the license issue, but the license wouldn't be sufficient. On the other hand, the manpages *would* be sufficient, although you'd be violating a few people's copyrights.

Andrew: So what you're saying is that a key to running a good project—open source or otherwise—is to break it down into pieces that people can understand?

Tim: What I'm saying is that there's a framework in which you organize people's creativity. And there's a whole bunch of principles there. Let me give you a concrete story. Back in 1998, when I organized what came to be called the Open Source Summit, I had this plausible story about what was happening in the industry. There were other people talking about it, notably Eric Raymond, who'd written "The Cathedral and the Bazaar." I'd done the Perl conference the year before. We were all thinking about a bunch of things in parallel. Netscape had just open sourced their browser, so there was a lot of ferment. And I noticed something: the iconic story that was being put out by Eric Raymond and others left out the whole BSD world. It was all about Linux; it was all about GPL'd software; it was all about the ideals of free software.

Meanwhile, there was this whole other tradition that I looked at. And I looked at it and said, "Wow, that tradition has actually had more impact." So, for example, I'd go around and say to people, "Tell me the top five programs on the Internet." They'd scratch their heads. And I'd say, "Number one: BIND, the Berkeley Internet Name Daemon. Every one of you has a website that's depending on that program that's maintained by this long-haired programmer in Redwood City. Number two, either sendmail or Apache. Seventy-five percent of Net email is routed by sendmail." You went down the list, and it was all Berkeley software. People weren't seeing that story.

Jenny: I remember that at the time, there was a lot of controversy around choosing an open source license, GPL versus BSD. It sounds like you found a way to rise above that and bring everyone together. How did that happen?

Tim: I brought together all the GPL people and all the Berkeley people, and I said, "We have to find out what we have in common. We have to tell a story. We have a press conference at five o'clock. I don't know what we're going to tell them, but we're going to tell 'em something at five." I had a plausible idea that there was something really interesting going on here. But I didn't know that in the course of that day, wrestling through the issues, what that group of 20 people would do. Eric came in and said, "Well, we had this meeting a couple weeks ago, and Christine Peterson proposed this new name, 'Open Source'." Michael Tiemann of Cygnus says, "Well, we've been thinking about this, too. You know; the problem of the name 'free software'." And Linus Torvalds said, "I didn't realize that 'free' had two meanings in English." And Michael Tiemann says, "Well, we've been using the term 'sourceware'."

So we had a vote. And we said, "We're all going to agree on one name, and we're all going to start using it." We voted on "sourceware" versus "open source," and "open source" won. We had the press conference at five, and the rest is history.

Andrew: That's interesting, since Jenny and I have written about how setting artificial deadlines can have negative consequences for teams. It seems like it really helped you guys.

Tim: A lot of what happened was that we brought together a very short-term group of people. I had a sense that there was something there, but didn't quite know what form it would take or ought to take. But I knew I wanted to make something happen. So one of the team principles is that creating an artificial deadline—"We've got a press conference at five o'clock, guys"—can be a pretty powerful tool. Now, a lot of companies and a lot of people really misuse that. I've seen situations where project managers lie to their teams about deadlines, trying to create an artificial sense of urgency. That violates trust.

But when you do it right, identifying that sense of urgency can be really good. It's a little bit like what Alexander Pope said about writing poetry in rhyming couplets: that the narrow aperture makes creativity shoot out like water from a fountain.

In terms of teams, the other thing that I would just really observe is the power of having people with complementary strengths. Right now at O'Reilly, I have a very strong chief operating officer, who makes a huge difference. We're kind of two halves of a coin. I'm the big-scale storytelling, and I spend much less time on internal minutiae of the business, which frees me up to do a lot more with that. She's all over the day-to-day running of the business, and doing a better job of it than when I was trying to do both jobs. Understanding complementary strengths is really critical.

Andrew: I've got a question. Awhile back, Jenny and I wrote a piece for ONLamp called "What Corporate Projects Should Learn from Open Source." And subsequently, we've done a talk called "What Makes Open Source Work." And what we focused on is the

practices they use, because that's been our "thing" in the past. You've been talking a lot about the team, the personalities, complementary strengths, how to work with the people. But one of the things we've found is that if you look at the most successful open source projects that we all know and love—like Apache, Linux, Emacs—you find a lot of great practices. The developers, the programmers, the people on the projects voluntarily adopt practices that, if they faced in the office, they'd find stifling, even though they're the same practices. Practices like adopting a very strict build and release process, continuous integration, test-driven development.

Jenny: Lots and lots of code reviews. We've actually been tasked with putting those same practices in place in corporate environments and gotten lots of pushback from the very same people. So why do you think that is?

Tim: Well, let me put it this way. Anyone who's ridden a horse knows that the secret of success is to have the horse think that it's doing what it wants. So I think that when people feel like somebody else is telling them what to do, there's going to be resistance. If they think it's what they themselves want, then they sign right up.

Lao Tzu said this 2,500 years ago: when the best leader leads, the people say we did it ourselves.

When some people create something really wonderful with an aesthetic vision, it seems really obvious. Now that we have the iPod, how could you imagine not having touchscreen devices? I remember the first time I picked up the Kindle. I started stroking the screen and nothing happened, and it was like, "What's wrong?" Now the Kindle has its own pieces of truth in it, so to speak—like the EVDO connectivity, where you go, "Yes, that's how it's supposed to work."

Andrew: I think it's interesting that you're talking about how a product's supposed to work. Most of the discussions I've had about open source are about things like the GPL versus the BSD license versus Creative Commons, and I think people tend to get a little bogged down in those details. But you're talking about open source software as a unique way of realizing a vision, and I've never actually thought of it that way.

Tim: Well, you have to realize that people always get hung up on licensing as critical to open source. And while it is certainly important, I think as you guys have suggested here, the practices are much, much more important. When I think about what we've done in our business, first of all books have always been user-contributed. Most of our products are created by people who don't work for us. We have to put out a plausible idea of something we want to get done. We have to find someone who says, "Yeah, that's a good idea, I want that, too." We give them coaching, we bring in a host of other people to review their work inside the company and outside. We have to impose a management system, although it's often very loose. There are a lot of different ways to do it, which is one of the reasons I love Larry Wall's Perl slogan, "There's more than one way to do it." Because I don't think there's any one answer.

Andrew: Especially not in Perl.

Tim: Yes. And take the book *Programming Perl*. I could spend six months remaking this book to be the kind of book I'd write. Or I could say, "Wow, it's really good just the way it is." Even though it's not the kind of book I'd have done myself. So I blessed it and sent it through.

I remember this one book we published. It's really unfortunate because the editor was really fixated on making it into a different kind of book, so she spent two years working with the authors to somehow teach them, force them, cajole them into doing something other than what they were originally imagining. She just couldn't let go. So I said, "There's a book here, but it's a different one than you're imagining." What these guys had done was written a whole bunch of disconnected pieces. I said, "All you have to do now is tell the story of the book so that this makes sense."

Jenny: I've had a lot of people on my teams tell me that Programming Perl is one of the best programming books ever written. You had an expectation of what the book was going to be, but what you were given turned out to be different—and better.

Tim: That's sort of another piece to all of this. How do you find what's true in what you're given? I think this is almost the root of intelligence that draws everything else. There are different kinds of intelligence. Once kind is essentially algorithmic and manipulative: you're given all of this data, and you're good at manipulating it. You see people who seem like they're really bright in some sense, but they're really dumb in other ways. Like they don't see things that are obvious, they don't have any common sense. Then you see somebody else, and they're not that good at symbol manipulation, can't spell, and can't do math. But they're geniuses at looking at a situation and understanding what's really going on. And the smartest people have both of those qualities. They can look at the world fresh; they can look at something and say, "Wow, I see what this can be."

So going back to that other book, the editor had all of this training, all of these ideas about how a book should look, and what should be happening here. And that got in the way of her being able to see the material with fresh eyes and understand that the process could be better by just helping the authors to do what they wanted anyway.

Jenny: Do you think that an editor—or any team leader, really—should try not to have a strong hand in pushing the authors (or the team) toward a specific goal?

Tim: Most of my experience is very much being a leader, not necessarily being a team member. A great deal of it comes from directing people, and I try to direct them in such a way that I have as little to do as possible. A lot of that requires seeing people's strengths, seeing a situation and saying, "Here's what we can make of it."

It's a kind of pattern recognition, which is going back to how I think about the process of editing. It's a little bit like what Michelangelo used to say about making a statue; that it's about finding the image that's hidden in the stone. I think editing a book is like that. Leading a project is like that. When I started telling the story about Web 2.0, it was looking at a bunch of data and uncovering the statue that was hidden in the stone. Same thing when I

told the story about open source. I think that leading your team is like that also. How do you get a group of people to achieve their potential? By seeing who they are, and what they can accomplish.

Andrew: Do you think it's possible to have a great team that doesn't have a great leader? That has more of a collective leadership?

Tim: Yes, it is possible. But here's the thing. Take Apache, because I think Apache is the great example of that. Tim Berners-Lee laid down the blueprint. He said, "I've created this idea for this hypertext server, this hypertext client." And the genius of Apache was in embracing the constraints. I still remember back in the mid-'90s, this moment where Netscape had added this, Microsoft had added that, and everyone was saying, "Apache seems to be standing still. They aren't adding all these features. They aren't keeping up!" And the guys at Apache said, "Yup. What we do is a hypertext server, and we have this nice extension mechanism where people who want to do something else can add it on."

And that goes back to that architecture of participation. They didn't build this big, conglomerate, complex application. They kept to a pure vision. The vision did actually come from a visionary leader; it just wasn't part of Apache. Apache came from a group of people who were abandoned by the NCSA server team when they all went to found Netscape. And there were a bunch of customers, so they said, "We have to maintain this, and keep it going." What was wonderful about that kind of team was that they accepted the constraints that were laid down by the design of the system. They didn't try to show off their ego or their creativity.

I think a lot of the work done by the IETF (the Internet Engineering Task Force) in the early years did the same thing. There were some wonderful principles laid down, and people really honored them. If you read some of John Postel's stuff in the TCP RFC about the robustness principle, it sounds like something out of the Bible, for Christ's sake! "Be conservative in what you do; be liberal in what you accept from others." Literally, that's what it says.

The point is that if you have the system architected right, you have a better chance of success for teams. You don't want teams that are dependent on a single vision or leader, because if you lose your leader the whole team goes "pop".

Andrew: Speaking of things going "pop", can you think of anything from your past where things didn't go so well? Maybe a disaster or two that maybe you learned something important from?

Tim: There are probably quite a few disasters over the years, some of them which were turned into successes after the fact.

These aren't disasters, but they are failures—they're choices, going back to an aspect of leadership. Things can go wrong. Look at what happened here with Yahoo! and Microsoft. Everybody's out there saying that they can't make up their minds whether Jerry Yang or Steve Ballmer is the bigger loser, because there was a badly mismanaged process without a clear vision of what was the right thing to do. But we might end up looking back and say-

ing, "Wow, Jerry was brilliant. He kept Yahoo! independent, and then he had a strategy that pulled his hat out of the fire." Can you imagine? Just like you look back at the early history of the PC, at the deal that Bill Gates had cut with IBM. What if IBM had been the sharp dealer? We'd have a very different history. So we don't really know.

Jenny: Right, what we think of as a set-in-stone success now might not even have seemed like a wise choice at the time. You make a series of choices, and the chips fall where they may. Is that how you see your own successes and failures?

Tim: I see in those kinds of moments the impact of choices. At O'Reilly, we did the first web portal. We did GNN, the Global Network Navigator, the pre-Yahoo!, the original portal of Internet sites. I was focused and I did not want to give up control of my company, so we sold GNN to AOL. I did reason correctly that unless I wanted to give up control of my company and take on investors, I wouldn't be able to keep up with the growth of the Internet. I'd read this wonderful book called *Marketing High Technology* by William Davidow, an early venture capitalist. He said that it's simple math. Dominating a market means being more than half the market, and growing faster than the market as a whole. I looked at the Internet and said, "We can't do that, a private company that's completely self-funded. With the way that the Internet's exploding, there's no way we won't get marginalized. We have to either take in money or sell." So I sold. It was a big inexperience premium, so to speak. Jerry and David took in venture money, and went on to create a multibillion-dollar company despite the recent travails. I think, from a purely financial point of view, they did much better than we did. That being said, I was clear what my personal goals were: to keep an independent company where I could do what I wanted, and I've been doing it ever since.

I guess the point I'd make is that failure and success are relative to what you're trying to accomplish. If we'd been a venture-backed start-up, my decision would have been a disastrous failure. As a self-funded entrepreneur who was making my own choices, it was my choice to make or not. What I'm trying to get at here is that it's really important not to second-guess a "failure," because a failure may be a choice.

Andrew: But don't you need to have some sort of final accounting? In the end, a project either succeeds or it fails, right?

Tim: We have this idea that's very binary, that it's failure or success. We're trying to choose from a set of alternative futures. There's no best, there's just choice.

I think we've lost in many cases, both in business and in the design of software—the design of any work product, really—the importance of the aesthetic. Wallace Stevens wrote a book of essays called *The Necessary Angel*. In it, one of the points he made was that we have the idea that choice matters. He had a poem called "Notes towards a Supreme Fiction." He had the idea that perhaps God was a fiction that we can all believe in. The goal in religion or even in science is to create an aesthetic vision that we can each believe in. You're trying to enroll people.

Andrew: I can hear a reader who works in a corporate environment building a database application reading that and thinking, "There's no aesthetic in what I'm doing."

Tim: And I would say, "Go look at Steve Jobs." That'd be my answer—he's a testament to the power of the aesthetic. He made a set of choices. For many, they feel that Apple screwed up because they didn't go with the dominant paradigm. They followed their own way, and their own aesthetic vision. Each time, he's been able to come back again and again because he has a compelling vision that he's been able to sell to people. People talk about the "Steve Jobs reality distortion field"—that's what it's all about. He can create a compelling vision that other people sign up for.

I remember one of my key employees laughing at me, saying, "I'd have a meeting with you, and I'd go away and I've said 'yes' to all these things you've said. Then I go away and think, 'I don't actually believe all that stuff!'" I was able to persuade her. And that goes back to what may be an interesting observation. I remember when my company got beyond about fifty people. There was this moment when I really had to change how I worked, because I really had this experience in the early years of the company where I held people in my "reality distortion field," so to speak—everybody in the company at the same time. It was a small enough group and we were all working closely together.

I remember when I was in high school I used to sneak out at night in my father's car. So he wouldn't be able to hear, I'd have to push it out of the driveway and down the block before I started the engine. So I'd be pushing the car, and there'd be this feeling that you're pushing on this thing, and you're pushing and pushing, and gradually it starts to accelerate. I had that feeling with the people in the company—"Wow, this thing is just too heavy." So I had to let go. Then I would talk to a few people at a time. You'd set people free to go do their thing, and then you'd check in.

There's a science fiction book I read early in my career that was very influential, a book called *Rissa Kerguelen* by F. M. Busby. Not that many people have read it. One of the key concepts that many people have played with over the years, particularly in the '50s and '60s (this book was written in the '70s), was the idea of time dilation. As you get close to the speed of light, one of the Einsteinian paradoxes is that the inertial frame is different and time goes much more slowly for the person traveling at high speed. Many science fiction stories would involve people going off and coming back, and everybody's very old. So this book, *Rissa Kerguelen*, had three parts, and one of them was called "The Long View." It was about how you do planning when you're about to go on an interstellar voyage, and you're going to show up 15 years later by planetary time. The idea is that you have to set something in motion, and then meet up with it. And I think there's something very powerful in that image, the fact that the things we were talking about earlier—the architecture of various systems—that's a way you set something in motion. And you can rendezvous with it, and find if it's developed in the way you expected. That was how I began to think about what I had to do in the company. I had to set things in motion and then go meet up with them.

Jenny: How did your team initially react when you started pulling away?

Tim: There are definitely cycles where people feel their own oats. They want to be the leader, and they want to take things in a different direction. And I think one thing about being a good leader is to know when somebody has the chops to do that. I'm a Celtics fan, and I remember there was a great story from the Larry Bird era. It was crunch time at the end of a basketball game—this was when K. C. Jones was coach. They'd come to the huddle, and Larry says, "Just give me the ball and everybody get out of the way." And K. C. says, "Larry, I'm the coach, so shut up! OK, everybody? Let's get Larry the ball, and everybody get out of the way."

So sometimes people move ahead, and your job as coach or leader is to say, "Yeah, they're right, give it to them." I've had that relationship with Dale Dougherty a lot. Dale is actually responsible for a big part of O'Reilly's success. A great many things that are attributed to me, he played a major role in. He was the guy who originally got us into the World Wide Web. He was the guy who came up with the name "Web 2.0." He's currently the publisher of *MAKE Magazine*. It kind of feels like a dance in which he goes off and does his own thing, and then it comes back together. Sometimes it feels like a sibling relationship, where we're struggling over who's driving the bus. Other times we're really in harmony. You want people who will argue with you. You want people who have their own vision.

When you have a vision of something that is true—and I don't know what true means, if it's absolute truth or just aesthetic truth. When things just work, when you hear that perfect chord in music, when you see the line in a drawing or the curve of stone in a statue, and you go, "Beauty is truth, and truth beauty, that is all ye know on earth, and all ye need to know"—you know, the Keats line—that's how things come together. For me, the essence of getting people to work together is to have an aesthetic vision that you can get them to sign up for. Where you build a shared vision of the truth that you're building, where you've expressed an ideal. Because then you set people free to pursue that ideal on their own.