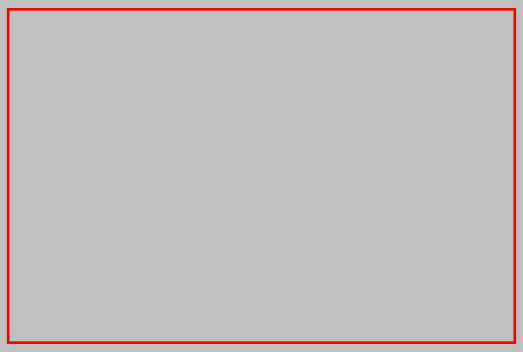
The New Age of Infoware: Open Source and the Web

Tim O'Reilly



RealVideo: Modem | ISDN

I hope you don't mind if I wander around out here, rather than standing behind the podium.

I'm gonna talk about something perhaps a little different than you are accustomed to hear at a meeting about open source. Very often when people talk and think about open source, they first go and think about Linux, the GNU tools and the whole operating system layer. I like to bring peoples' attention to the role of open source in the web.

I believe that the web is changing the way that we use computers and the way we're delivering applications to people. Understanding the role of open source in the web gives us some very different perspectives that I think are very, very important for the open source community. So in order to give you this perspective, I'm really gonna go through three main points. First, I'm going to give you a slightly skewed history of the recent computer industry, the last ten, fifteen, twenty years or so. And after that, I'm gonna talk about how people make money with open source, and how people are making money in general in the computer industry. And then finally I'm going to come back to this idea of how the open source community needs to deal with the web and think about the impact of the web on open source in the future.

So let me start first with this idea of revisionist history. It's a little bit of a personal history because that's how I, for example, decide what books to publish. It's the things that interest me and the perspectives that hit me upside the head as I go through life. So when I first came into the computer industry in the late seventies it was a hardware dominated industry. The companies that I dealt with were what I would say 'computer companies.' They weren't hardware companies or software companies, we didn't really have this distinction in one sense. Sure, there were some companies devoted to software, but mostly they were satellites of hardware companies in one way or another.

So for example, when I started working I did a lot of work for Digital [Equipment Corporation] and then Hewlett-Packard and Data General. So the hardware companies dominated the industry. Something happened in the early eighties that was fairly revolutionary, and it had enormous impact for many years to come. And that was a decision by IBM, a decision that was part accident, part inspiration by some rebels in the organization who kind of slipped it by their top management, partly perhaps a stroke of genius, partly perhaps an accident. And that was the decision to release the specification for the IBM-PC as a relatively open hardware specification. IBM said, 'Gee, anybody can make these things.' What happened was that the market broke wide open in a variety of ways.

First of all you had a whole lot of new hardware companies, and those companies no longer had to invent new hardware in quite the same way. Those of you who may have read a book that was popular in the eighties *The Soul of a New Machine* [by *Tracy Kidder] which was about the design of a machine at Data General, and remember that was one of the big frontiers in the computer industry, you know, if you were at Data General, the way you got ahead was to design a new piece of hardware.

Now once IBM released the specs for the IBM-PC, everything changed. All of a sudden, and this is the most important thing to me, there was what I would call a 'loose coupling' between hardware and software. You no longer had to be a hardware company or a satellite of a hardware company to be a player in the computer industry. So for example, I knew Mitch Kapor, the founder of Lotus, before he started Lotus, he was actually studying with my wife, he was thinking of becoming a family therapist. And he was sort of wondering, should he spend time getting his masters degree in family therapy, or should he do something in the computer industry? So he really didn't have a whole lot of resources, but he wrote Lotus 1-2-3, started Lotus and became quite wealthy and changed the industry a great deal as a result. So all of a sudden, here was just this guy off the street, effectively who built an enormous company that had an enormous impact. So that loose coupling, the openness that was behind the PC, changed the rules.

Lemme talk about some other ways that it changed the rules, and this is very very important, I'll come back to this later in talking about open source. You look at a company like Dell. Michael Dell, in his just recently released book, actually talks about an interesting thing that happened with Dell. They were in the business just like everyone else thinking that this was really about inventing innovative hardware, and they had some big project that was getting more and more behind and was really causing them a lot of heartburn as a company. They decided to cancel the project. And in retrospect Dell said that was a critical turning point for the company because they realized that in the end their point of competition was not going and inventing some new hardware edge, it was in getting better at sales, marketing and

distribution of what was essentially a commodity product. And if you look for example, at the growth trajectory of Compaq vs. Dell, that realization of the impact of the paradigm shift is extremely important.

So now let me fast forward a few years to the late eighties. Those of you who remember my books on the X-window system, remember they were shipped by a whole lot of Unix workstation vendors. And all of these vendors came to us with a story, they said, 'We believe, that in the future we will be delivering documentation online.' And they said to us, 'we have this great idea, we want you to put your books into our specialized format.' So Sun said, 'just put your book in AnswerBook, and won't that be great?' The only problem was that IBM was telling us, 'just put them in InfoExplorer, won't that be great?' And there were various software suppliers who had online book-reading software and before long, we realized that if we wanted to publish our books online, we would have to have thirty different formats.

So as a result, we started a group called the Davenport Group, which was a working group to establish an SGML *DTD for online manuals. It was called DotBook, that's since been adopted by a whole lot of Unix workstation vendors, it's also now been adopted by the Linux community. But we did that, because we felt that we needed to somehow break this coupling between information and a particular software package. This was very interesting, because just as in the seventies, you could only do software if you made a choice of hardware, if you are a satellite to a particular hardware company, in the late eighties you could only do online information products by tying them up with a particular software company. And we didn't want to make that choice, we wanted to break the coupling.

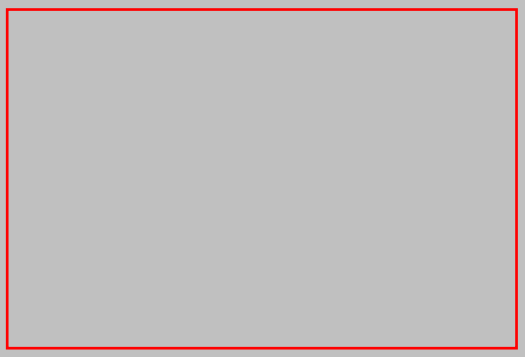
So in addition to working on DotBook, we also looked for what we could distribute as a free browser. We actually worked with a guy named Pei Wei at the University of California at Berkeley who had a product called Viola, which was sort of a HyperCard clone, ran on Unix, and was a tool kit for building hyper-text applications. One of those applications was actually the first graphical web browser. So we started working with Pei because we felt, well we want to have this free book reader to go out with our books. I kind of laugh, cause I later told Pei, 'Gosh I wish you'd been discovered by a software company you might have been the foundation of Mosaic, or whatever.' Shortly thereafter the NCSA guys started working on Mosaic, actually, according to some reports, inspired by a piece of junk-mail that we sent out advertising a web-product that we had developed, the Global Network Navigator.

But the point was that we were looking, and some of the origins of the web really came from this idea to break the tie between software and information. Now what was really interesting, was that in the early days of the internet, this choice that had first come up for us with our online books, in the late eighties, was even more

apparent. We had Microsoft come to us, and they said, 'We have this great vision of the online future, there's gonna be this incredible network for information delivery, it's called 'The Microsoft Network." Nathan Myhrvold gave a talk and it was a great talk actually, you know he had this wonderful vision about how online information would change the way people communicated, the way they related. The only problem, I went up to him after his talk and I said, 'Nathan, you're talking about the internet.' And of course because Microsoft hadn't yet had their internet realization he said, 'nahhh.' But you know, six months later they realized it was the internet.

But what was interesting about this particular interaction was that you had a choice and behind door number one, it was 'here's our 15 page contract,' or it was a 20 or 25 page contract from Microsoft, and it had things like 'pay an entry fee of $50,000 to be a content partner' and 'give us a share of any revenue you get.' And over here behind door number two was the world wide web that was starting to gain steam, and had an open standard, anybody could play. You could build anything you liked. Uh, which did we choose?

So once again you had this decoupling of things that had been tightly connected before. Just like you had this coupling between hardware and software, now you have this coupling between software and information products. The web broke that, broke it wide open. And this led to a real paradigm shift in the industry. And that's something that I think we are just starting to realize.



That paradigm shift I find is best illustrated by a small story about Amazon.com. Everytime I talk about Amazon, all the booksellers who resell our books kind of throw daggers at me because they don't like me talking about them. But I really do think they are one of the bellweather applications of the decade. And notice I said 'applications.' And that's really the critical thing. This little story I like to tell is of a friend of mine who doesn't currently own a computer, and who told me one day that she was thinking of buying a computer so she could use Amazon. And the reason I thought that was so interesting was because, back in the early days of the PC, that was what people said about spreadsheets, in fact that was the definition of what they called a 'killer application'. You know, it was an application that made people go out and buy a computer.

So when I heard this person, who's a massage therapist, saying she was going to buy a computer so she could use Amazon, the words 'killer application' went off in my head. And so, I started thinking about Amazon and Yahoo as applications, not as web sites. A web site was something you 'published' and it was all about documents

that were put up on line. But applications are about things that people do. And of course if you think about that, if you think about Amazon as an application, you realize that it's a very very different kind of application than Microsoft Word or GNU Emacs. I've started to call this kind of application an *Infoware application* rather than a software application. Because at bottom the user interface is not built with traditional software components. To be sure, there's a great deal of software in it, just like there is a great deal of hardware underneath the software applications that have taken over. But this new layer is not strictly a software layer, and this realization, I think, is very, very important for developers to think about.

A lot of this thinking really came from observations that I made from my publishing program. And a lot of it started to crystallize really in 1996, when I was watching... in '96 and early '97. In all the newspapers, in Infoworld, in PCWeek, places like that, they kept talking about ActiveX, and you couldn't turn a page without coming across a story about ActiveX. And we sold a lot of books, and we saw people buying Perl Perl Perl. We didn't publish any books on ActiveX but we knew a lot of other publishers that did and they'd put them out there in the store and they'd kinda lay there. And so we knew from some kind of ground level that there was something badly wrong with ActiveX.

Now it might have been that it was just bad software, but I went further than that in my thinking. I started to say, I think that there's a paradigm shift at work, because what I saw happening with ActiveX was that it was an attempt -- and actually Java applets were the same -- to turn the web back into a software product instead of an information or an *infoware* product, because what was so critical to the web was this idea of the low barriers to entry. You know, HTML, people don't really think about this because, of course, we are focussed on software and programming, but HTML is one of the great open revolutions, I mean anybody can build an HTML interface. Again you don't think of it as source code, but just imagine for example, the Mosaic and then the Netscape Browsers had not included a 'View Source' button. Would the web have taken off the way that it did? People copied HTML pages, they built on them. Perl had an incredibly important role in the same way, people could download a perl script that was behind somebody's cgi-bin directory and they could modify it. So it was this idea that you could do it by imitation, you could just pick up on what somebody had done and run with it. It has a lot of the characteristics of the open source community.

You notice I'm skirting the strict definition of open source because while I think licenses are very, very important, what's more important is what people do in practice: Do they share? Do they copy? Is it easy to build on what other people do? And I think the web was incredibly driven by those low barriers to entry that were implicit in having that HTML 'View Source' available all the time. And what's more, you didn't have to be a programmer to make a web page, you didn't have to be

a programmer to build an application, we really started to have this new layer on top of the software. Again, there's a lot of serious development behind a major web site, I don't want to minimize that. But at the same time, there are many many things about web interfaces that are different.

I like to compare and contrast an application like Amazon with an application like Microsoft Word. One thing I like to say is, a traditional software application is a lot of software with little bits of embedded information. You know, you've got those silly pull-down menus, you've got a little bit of embedded help, little tool-tips buttons that pop up. The information is embedded in a lot of software, and of course, if you look at the bloatware that Microsoft delivers, it's a *lot* of software. I believe that, by one estimate, there is more software in Windows 95, an order of magnitude more, maybe several orders of magnitude more than they used to send a man to the moon. On the other hand, web pages and web applications you can almost look at as little bits of software embedded in a lot of information. And so for example, the interface to Amazon is built dynamically of millions of pages of text, and they've got these little links in them that tie back to the software layer.

So this is a different paradigm, and one of the things that I spent a lot of time thinking about was why we kept seeing this incredible strength from our publishing program in scripting languages, Perl, to a lesser extent Tcl and Python. They seemed to have enormous impact, so I spent time thinking about why that could be. And so I talked to people, for example, to Jeff Friedl, who some of you may know as the author of our book *Mastering Regular Expressions*. He works full time as a Perl programmer at Yahoo, and I said 'Jeff, what do you do, at Yahoo, with Perl?' And he said, 'well you have to realize that what we're doing at Yahoo is building an application that is happening in real time, all the time.' He said that he actually works for quotes.yahoo.com and his job is to write humongous Perl regular expressions that look at all the new feeds, Reuters and PR Newswire, all the online information that's kind of coming at you like drinking from a firehose, and he needs to figure out, out of all that text that's out there, which stories relate to particular stock tickers and then he's got to map it into the Yahoo stock space. So that for example, when you click on the ticker for IBM, you get all the latest stories. That's happening *live* all the time. It's software as a process much more than as a product.

And it's sort of interesting, I love the phrase 'Perl is the duct tape of the internet' because it suggests some of the ways that scripting languages are used in web sites. I look at this wire here, what do we have taping it down? Something like duct tape. Because whenever you're putting something up and taking it down quickly, you use duct tape. You know you use something that is a temporary, quick hack. So these scripting languages have not just an incidental role, but a critical role in some ways in these web sites because the application that they are delivering has a lot of dynamic qualities to it, it's happening *live*. So anyway, there are a variety of things

that make web applications different, I think we are still thinking about all of them.

But I just wanted to set that stage -- and I know it's a long stage setting -- for thinking about what this might mean for the web and for the open source community, so let me move to my second topic. And that is when you have one of these transitions, in which we move from a tight coupling between two layers in the computer industry to a loose coupling of some kind, and you can build on top of that standard. Let's think about what I called the hardware to software transition, kind of like the cretaceous boundary or whenever it was that the dinosaurs got killed off. What happened there was that there were a whole lot of people who figured out the rules in a different way.

Now, one of the things that we are all very very aware of is that there was an enormous shift of power from IBM as the 800 pound gorilla of the computer industry to Microsoft as the 800 pound gorilla. So there was a shift in power, or at least in perceived power, from hardware to software. Secondly, as I mentioned, there were some unique niches exploited by people like Dell who realized that there's different business models when you have commodity hardware. But there's also some other interesting observations that you can make. Who else besides Microsoft was a huge winner in that hardware to software transition? Well of course, Intel. And what I think is very, very interesting about Intel is that they realized that if they could figure out a way to get a proprietary plug in to this open infrastructure they could make a heck of a lot of money. And in fact, they did, because every one of those open architecture PCs ended up with an Intel processor chip. Okay, so that tells us something interesting that we have to keep in our minds going forward. So let's think about the web revolution: is there an analogue to Intel? You got it: Cisco. That's exactly right, here the internet took off, open, open protocols, open connectivity, somebody figured out 'ah, we have this nice little proprietary layer that we can kind of stick in there and leverage the hell out of...' Cisco is the Intel of the internet revolution.

Okay, now let's think about this infoware revolution that I'm talking about. Who is the Intel, or the Cisco of the web? I don't think we know yet, but I think there is definitely some food for thought, because I think that there's a real danger there. And there's a real interesting implication for the open source community, because one of the things that is very very interesting is that open source has had an enormous impact on the web, and it's become kind of a truism, at least, I've said it enough times and maybe people have heard it, that for example, a site like Yahoo is built very largely on open source. They use the FreeBSD operating system, they use the Apache web-server, they're a very heavy user of perl, they also use Tcl, -- they're an open source shop. In fact, the very first time I gave a version of this talk was back in Wurtzburg at the same Linux Congress where Eric Raymond first gave 'The Cathedral and the Bazaar' and I made some of these observations, not quite so

well developed, but I made sort of the boastful claim that 'open source is the 'Intel inside' of the next generation of computer applications.'

But as I think about it, I think I spoke too soon, because open source is right now the inside of that next generation. But I don't think that there's a guarantee that it will stay that way. That's maybe the real point of my talk, because it seems to me that if you analyze where the industry is going, I see the open source community is very much focussed on winning the last war. We're really focussed on the battle with Microsoft. Again back in Wurtzburg, we were gonna take the desktop; and now it's: 'well, we're taking the server and maybe we'll take the desktop next,' but still the focus is on beating Microsoft. And while I think that is a worthy goal, and Linux is doing pretty well at it, what really concerns me is that I see Microsoft focussed pretty heavily on being the Intel Inside of that next generation of computer applications. Much more clearly, strategically focussed on it than the open source community, which worries me.

Bill Gates got up there back in '97 and said, Netscape isn't our real competitor, Apache is. And that was great for those of us in the open source community because we said 'Yeah, we're on the map!' But think about that, I mean, Gates saw pretty clearly, that Apache was a clear target. Similarly, I made fun of ActiveX earlier where it was an attempt to professionalize, to turn the web back into a software product, which was then rejected by this sort of HTML community, but what did Microsoft do? They went back to the drawing board and they produced ASP. And they produced environments which were much more compatible to the scripting crowd. And if you go out there and do an Alta Vista search on files ending with .asp vs. .pl you'll see that they are making pretty good progress. They are attacking the open source community where it really counts and believe me, for that matter, they are attacking other parts of the internet as well. Exchange Server is targeted pretty clearly at the market that sendmail now dominates.

In fact that's one of the reasons why Eric Almond felt like he had to go commercial, he had to get additional resources to compete. And he asked that question, 'Do you really believe that smtp will remain an open standard if Microsoft Exchange becomes the dominant mailserver on the internet?' I don't believe so, and I don't think you'll see open mail standards at all. You might have MSmtp, you might have MPOP, MIMAP, and of course these things will only work with Microsoft client products, so there's some real important issues here. Will open source continue to be the dominant standard enforcement, if you like, of the internet? Both Eric Almond with sendmail and the folks from the Apache group are really both experiencing the brunt of this atack, and feel very very clear in their minds that without their work, without having their dominant market share, that those protocol standards would be going the way of all flesh or the way of all history, I don't know, they would be no more. So I think there are some very important battlegrounds.

There's a further important battleground, and this goes back to my infoware story. If you think about Amazon as an application there are several implications there. One is, you know, what's the open and closed story about someone like Amazon or Yahoo? Yeah, they use free software, Amazon does as well as Yahoo, and that's pretty cool. But if they are an application, first of all several things are pretty serious implications. One, is that the whole way that we are used to thinking about software and software distribution is based on the idea of software being passed from hand to hand, but when somebody uses the Amazon infoware system, they actually use it in kind of a time-sharing way. They don't actually have to distribute any software, so they could build all kinds of cool, interesting, perhaps even dominant proprietary things, on top of the open software architecture of the internet, and all of our vaunted open source licenses wouldn't even touch them.

Now I'm not saying that they are intent on doing these things, but I don't think they are thinking about it, and I think it's very very important to get people thinking early before there is too much money at stake. Because if you don't form good habits early, it's a lot harder to form them later. And going back to my earlier story about this open PC hardware standard, on top of that open PC hardware standard, we ended up with some really closed software standards, so openness doesn't mean much when you are making one of these transitions to a new layer. And I think we really are at that transition point. So, for example, what I think is very very important in this new transition layer will be what you might think of as high level protocols for communication between cooperating computers on the internet.

Right now we kind of have low level protocols like TCP/IP, we have medium level protocols like HTTP, on top of that we're gonna have various kinds of XML based data-exchange protocols. The question is, who's gonna control those, and when are the companies that maybe have dominant market share, gonna realize that they may have the world by the cojones, just like Microsoft did in the PC space. A good example of this from my business, we have to provide electronic catalog information to all the online booksellers. Well, our webmaster's having conversations with Barnes and Noble.com and Amazon.com and Borders.com and all the independent bookstores who want this same kind of information, and they all have different formats. And you know, Alan, the web master is trying to get them to agree, but if they don't agree, whose format do you think he's gonna implement first? Amazon, right, because they're the 800 pound Gorilla.

Now again, I don't wanna say anything about Amazon's intentions, I think they mean well, but the fact is, once a company gets enormous market power, its very easy to slip into habits of abuse. I don't even know that even Microsoft started out from in the beginning wanting to get everybody by the short hairs, maybe they did, I don't know, but once you get in that position, it takes a lot of strength of will and

strength of purpose and high moral fiber not to take advantage of it. So part of what I'm trying to encourage companies like Amazon and Yahoo to do is to think further ahead, to think about their responsibilities to the internet and to the future, because I think that all of these companies have benefitted enormously from the openness that the internet has sprung from.

I think the open source community, and what I really think of, when I say 'open source', I go back from specific licenses to really this idea of collaborative wide-area computing. In some ways I think of the USENET as the mother of the opensource movement. I know that a lot of people think about it on a license basis, but I think at the end of the day, what's really important is sharing. That people find ways to work together collaboratively and that's what really brought us to where we are. And so I think it is very important from these companies who have taken this enormous benefit from the internet that kind of grew up under everybody's noses while Microsoft was working on such splendid innovations as Microsoft Bob. Instead this world-wide collaborative community came up with the web, it came up with email, and all the things that make the internet so interesting.

So, I think its very important for companies who are involved in building and using some of these new kinds of applications, to think, to get in good habits, habits of openness, habits of sharing. And so for example, I'm trying to get Amazon to start thinking about participation in standards for data interchange rather than just going their own way.

I want to take a little bit of a side tour, I want to talk a little bit more about money. I think its very very important for all of us to think about the impact of money on our industry. When people talk about who's made money off of open source, they will point to me and say, 'Ah yeah, you've made more money off open source than anyone, you just give away the software, and sell the manual.' And even though I actually have made quite a lot of money because of open source, I figure I probably sold more than a hundred million dollars worth of books over the years relating to open source, and I think I probably had more open source related revenue this past year than either SuSE or Red Hat. I still think that that's not at all a true statement. I am not the person who has made the most money off of open source. The person who has made the most money off of open source so far is Bill Gates.

And the reason is that he had this great development lab that brought him all this new functionality that he took into his products. So one way that the industry makes money is people figuring out how to play it into their existing business. I mentioned Microsoft Bob earlier, did people buy upgrades to Office97 and Windows 98, all those billions of dollars of upgrade fees, for Microsoft Bob? No they shelled out all that money for internet functionality. They shelled it out for the functionality that had been developed over a period of fifteen or twenty years by a bunch of

collaborating researchers, hackers, people who just sort of built tools for their own use. So, the people who profit aren't necessarily the people who build the software.

We have to think about that, and we have to think about how do we make sure that we create business opportunities that encourage the further development of free software, of open source software, of software that really matters. I actually will make this case straight to Microsoft, in fact I'm going up there, I know Eric Raymond went there and talked to them last month, well I finally got an invitation, and I really think that they ought to be figuring out how they could give something back to this community, because they have profited immensely. I also think it's very important for these web companies. But it's interesting, it's really important to think about how people make money, because if you start to realize the impact, which is not necessarily implicit in your model -- if your model is wrong, you can just get blown out of the water by somebody who gets it right.

That was kind of my point in talking about Michael Dell. Compaq thought like Digitial: we're a hadware company we have to invent new hardware. Dell realized the business rules were different. Now I think Red Hat is doing something similar. Bob Young likes to say: 'We're not a software company, we're a sales, marketing and distribution company, just like Dell.' Now whether or not Red Hat in fact is the winner in what's shaping up to be the Linux distribution wars, and probably the next big Wall Street feeding frenzy is somewhat irrelevant. The fact is that there is a business model niche.

But let me talk about some other people who've made enormous money from open source software, and these are the ones that I find are really interesting. Another person who's been an enormous winner, Rick Adams, the founder of UUNET. How many of you remember when Rick was the hostmaster at *Sizemo (?), the world's largest Usenet hub, and the author of BNews, the most widely used USENET news software. Rick didn't say, 'I'm gonna put this software in a box and sell it,' what Rick did, when he saw that his bosses at the US Geological Society were starting to ask, 'Gee why are our phone bills several hundred thousand dollars a month for passing USENET feeds to anyone who asks,' and he realized that we needed to have some way for USENET to pay for itself. And he really invented what we now take for granted, the commercial Internet Service Provider Business. So when people think about free software and money, they think, they very often play right in to Bill Gates' hand because they think about the paradigm that he has perfected so well, which is put software in a box, ship it, get a locked in customer base, upgrade them. And Rick just kind of went sideways from that. And he was the first person to say, 'I'm gonna build a serious business that is based on free software,' and it was basically providing a service that was needed by the people who used that software, who talked to each other, who distributed it, who worked with each other online.

Now, I think the web companies are the next generation there, people like Yahoo. So what's really happening now in terms of the way that you often will make money with free software is that you actually deliver a service based on it. Either a service that underlays it, like the ISP service, or a service that's built on top of it like Amazon or Yahoo. Now if this is the case, then first of all we have a good answer for all these people saying 'this free software thing must be a bubble, because we can't figure out how anybody's gonna make money at it.' My argument is that people are already making more money at it than we can count. So that's a very, very important paradigm shift and it brings me back again to this idea of where the free software community, where the open source community should be focussing its energy. Because if the frontier is of developing applications that deliver online services to people and those applications are not distributed as applications, they are just software-in-use, then first of all, we know that free software has an economic model behind it, but we have to make sure that the people who are using that software realize it and that they, for example, contribute to the community, that they keep that engine going, that they continue to have that software that they are building on.

And secondly, in our development work we need to focus on the technologies that are needed for this next generation of applications. Now, I'm not saying that more traditional software is unimportant, far from it. Advances in operating systems are incredibly important, just like advances in hardware, and in fact in that previous transition, the hardware companies continued to do very well, there's been enormous growth in functionality, and I think there will be enormous growth of functionality in the software layer, but it's extremely important for us to not let the engines of the web go proprietary. It's very, very important for us to engage the companies that use web software in the open source community, in the open source effort, in open standards, because if they go proprietary, it won't really matter if Linux wins the server wars or the operating system wars. Because, well... the battleground is moving on. Anyway that's kind of what I really wanted to leave you with, I just want you to think bigger about the impact of open source. The enormous positive impact has been one of the engines of enormous change, and I think it will be the engine of even greater change as we go forward, as we see handheld devices start connecting, we'll see the cooperating servers. I really believe Sun's slogan is way ahead of it's time 'the Network is the Computer.' I think we're really heading into a very very important time and I think open source can be one of the key engines of that revolution if we focus energy in the right place, and that's what I would like you do. Thank you very much.

(Transcription Christopher M. Kelty)