

The Internet Operating System

Tim O'Reilly

Discovery Expert Series is a series of interviews by John Furier, the founder & former CEO of PodTech Network, Inc. John has a history with Internet companies serving as Vice President of the Product Group and Strategic Development for RealNames, an Internet search and naming company, and founder and CEO of Labrador Software, a paid keyword search and navigation company. John also has experience at large technology companies, having spent nine years at Hewlett-Packard in various technical, marketing, and sales roles.

Tim O'Reilly, a pioneer in web technology, discusses what it means for the Internet to be a platform. He proposes companies that value the user will be the real winners in the Web 2.0 and give sound advice on technologies and approaches that retailers and media companies should embark on to be successful in this new era of user driven content.

Tim O'Reilly is the founder and CEO of O'Reilly Media, thought by many to be the best computer book publisher in the world. The company also publishes online through the O'Reilly Network and hosts conferences on technology topics. Tim is an activist for open source, open standards, and sensible intellectual property laws. Since 1978, Tim has led the company's pursuit of its core goal: to be a catalyst for technology change by capturing and transmitting the knowledge of "alpha geeks" and other innovators.

[Edited from original transcript]

We're here with Tim O'Reilly, the founder and CEO of O'Reilly Media, one of the best computer book publishers in the world. Tim is most recently known as being a thought leader and visionary in the evolving modern web space and has coined the term "Web 2.0," and holds a conference. He really is leading the trend towards this modern web or semantic web environment.

Let me start by just clarifying that it wasn't I who coined the term "Web 2.0," it was Dale Dougherty who at the time was my VP of online initiatives, but is now the publisher of Make magazine which he launched.

But the term Web 2.0 was a new name for something I've actually been talking about since 1997, which is about the emergence of the Internet as the platform. We wrapped the name around my ideas. Before that I was calling it "The Internet Operating System." Web 2.0 just happened to be the name that stuck. Dale came up with the idea that we should launch a conference about the resurgence of the web after the dot com busts, and that's why we called it Web 2.0. "The web isn't gone; it's coming back." And then I saw that as a great opportunity to tell this bigger story that I'd been seeing and talking about: What were the rules for the new economy, so to speak.

The Internet Operating System

Tim O'Reilly

Discovery
Expert
Series

Let's talk about the Internet Operating System 'cause that really kind of hits at the core of really what's going on today. And Web 2.0 has become a punch line or a framework for what people can generalize it. But really, it's a changing environment. Web 1.0 was really just access the information—websites, etc. But the web is changing. Talk a little bit about where [the web] is today, where now everyone's connected.

Well I think that my actual thinking about Web 2.0 started with open source software. I was convinced pretty early on that the most important thing about open source was not the licenses; it was not even access to the source code. It was that open source developers were like the canary in the coal mine. They told us something about how online communities could be harnessed to build collective work. And it was effectively the first sign of a cultural revolution. And that was really where I started.

But the second thing I started noticing was that, when I started thinking about open source, the most important open source applications seemed to me to be ones that were outside the model of how the open source community thought about it. I eventually wrote a paper called, "The Open Source Paradigm Shift." I give this talk and always open with a trick question: "How many of you in the audience use Linux?"

I'd get a lot of hands at Linux Expo and a few hands at Microsoft, but I always got some hands. But it was never the same number of hands as I got when I asked my second question which was, "How many of you use Google?" Every hand went up. I said, "Well you guys just told me that you still think about what you have on your desktop as the operating system you use." And that's the heart of the open source paradigm shift.

What really happened is open source turned out to be this key driver in the move to server-based computing and cloud computing and the biggest and most popular things on Linux are not things like the GIMP or OpenOffice.org. They're applications like Google and Amazon. What does that mean? When the killer apps of this next generation of open source aren't open source, they don't even look like they run on your local platform. They run out there in the cloud.

And so I started thinking about that and talking about it. And then, of course, P2P came along and - you know, peer to peer file sharing, and that was telling us more about this future Internet. And then of course we had the dot com bust, which was obviously a fairly shattering experience for a lot of people. But what we noticed was that the companies that survived and the new breed of companies that were starting to come back after the die-off all had something in common: they were actually figuring out how to leverage the Internet in sort of a native way. They were really exploring the potential of the network to be the platform.

So this was the same story coming back again. And so as I started articulating that story. I said, "Well what do say Google and Amazon and eBay and Yahoo have that, say, Pets.com didn't," right? Pick one of the glorious successes of the dot com bust. You know, Pets.com thought it was about retail; they thought it was about television; they thought it was about eyeballs, whereas in every case of the companies that succeeded, they understood that users add value.

So eBay—pretty obvious. You know, a marketplace of buyers and sellers. Amazon—they really went out there and they worked their audience to contribute user reviews. They used collaborative filtering and similar technologies. In lots and lots of little ways they

wove users into the fabric of their product. Google—every time somebody makes a link, they actually are adding to Google. They're building their database. And Google is a real-time ad auction. Again, user-driven. Real-time services that are built on what the users are willing to pay for an ad right now. They don't sell it to the highest bidder. That was their breakthrough relative to the competition. They were able to actually instrument the site and in real time figure out what the market was. So that's harnessing users.

As part of that, I also realized that there's a shift in power from pure software to something – back in 1997. I called it “Infoware.” These applications are really driven by information. And one of the things that distinguishes a Web 2.0 application is it literally gets better the more people use it. And the reason is because it's a data-backed application, and the data is driven by user contribution, and the user contribution is driven by network effects. And that means the more people using the application, the more people contribute. And so you get this accelerating return relative to the competition.

That's why some other online bookstore just can't compete with Amazon because Amazon already has 10 or 15 million reviews. So you start and you have your little review thing and you get five reviews and, meanwhile, Amazon has 100. You want to set up a search engine and you have an ad auction but hey, why would they come to you when you only have a small number of customers, where Google already has tens of millions? So the companies that got there first and best and started to harness those network effects have accelerating returns that take them away from the competition.

And we see that pattern again and again in Web 2.0, that it's really the idea that if you understand the network and you learn how to leverage network effects and you build an application that gets better the more people use it, then you're doing Web 2.0. It has nothing to do with things like having a rich Internet application front end. It has to do with are you a networking application that gets better? And then again, there's also an aspect—are you real time? I gave a talk and I did a blog post at one point called, “What would Google do?” And I started off with a comparison of, say, a bank or a phone company and Google. I said, “Gee, they all have big data centers, they're all software as a service—they don't sell products, they sell services—they actually have databases that get better the more people use them.” I said, “But what's the difference?”

Well, the difference is Google takes that database and turns it into user-facing, self-service applications. The phone company, for example, has this back office where they know everybody who ever called, but they don't give it to you. You have the last ten people you called. If they were smart, they'd be Facebook because the phone is the ultimate social networking tool. Why aren't they exposing that database saying, “Here are the people you called. Now annotate it. Add value to the data that we have.” For example, if they built a creative address book application, they wouldn't have to have user-hostile service plans to keep people from switching because people would have so much invested in the service. They do a little bit of that with friends and family calling and those kind-of things, but there's so much more that could be done there.

I think that points to a future direction for many traditional businesses. If you're in business and you collect data and it's just in your back office, you're missing a trick. You have to think about how do you take that back office data, turn it into new, real-time, user-facing applications. And I noticed that Aggregate Knowledge is the sponsor of this podcast – Aggregate Knowledge is trying to do that, where they're basically trying to give you that, “people who bought this also bought that,” experience. But they're being even

more creative because they actually understood that some small, individual merchant may not have enough data to do that effectively, and they've figured out a way to do that in a cross-site database. And I think that's a really exciting part of Web 2.0. They went a little further in thinking about, "What does it mean for the Internet to be a platform?"

Amazon understood that, "Wow, if you can get all the users in one place and you get more of them commenting, you can get all these network effects." And these guys came along and said, "Oh you know, now everybody can do that, but if we could build an infrastructure layer for that to be shared, we can actually get to critical mass—that mass of activation of network effects—by a shared service.

Talk about software developers today—the Aggregate Knowledges of the world—people who are building solutions for businesses, and tie that in with users. So the software paradigm—you have the Internet and leveraging that as a native application or a native way or operating system; you have businesses who really need solutions to change their business. And you've been in the publishing business. But then you have the user behavior. And they're all kind of intersecting. Talk about the three dynamics.

Well, I guess I would say a couple things for software developers. And again, looking back with a historical view, a lot of my thinking about this actually started in 1996-97, when I was trying to understand why Perl was so successful. Our best-selling book back in the mid-nineties was programming Perl.

So I started asking developers, "What do you do with Perl?" And the breakthrough came when I talked to Jeffrey Friedl, who wrote a book about regular expressions for us, who was at Yahoo, and he said, "I spend my day writing regular expressions to match up news stories to ticker symbols." And I said, "Oh my God—Yahoo is an application that's never finished." It's an application where this guy has to keep doing what he does every day. Some new company gets created, it needs a new routine. A new set of key words. And that's actually where I first came up with the idea that Web 2.0 applications were effectively all Mechanical Turks. Jeff Bezos later took that term and gave it a really interesting twist.

I was using that, really, to describe this idea that Web 2.0 applications still have the developers hidden inside them. For those who don't know the story of the Mechanical Turk it was an automaton that purported to be a robotic chess player, which traveled around in the early 1800s. The secret was they had a man hidden inside. These applications only work because the people are still there.

And I remember talking with Marissa Mayer once at Google about this and I said, "If Google didn't keep doing what it does, it would be useless within a couple of weeks." And she said, "Oh no Tim, you're wrong. It would be a couple of hours." And that's why, for example, scripting languages are so important.

I used to refer to the phrase that was said about Perl—"Perl is the duct tape of the Internet." Well why do you use duct tape? You know, when you go to a conference, you're duct taping down your wires 'cause they're not gonna be there tomorrow. And so that's why we've focused on these rapid development environments as a key part of Web 2.0.

Yeah, it's hard-core infrastructure as well, but there's a lot of stuff where you're coding, almost in real time, you're trying features out. Do they work? No, take them down—try

something else, you know, versus say a software artifact like Microsoft Word where you accumulate cruft. You think about – features never go away from an old PC generation software program, they just kinda keep adding stuff.

You know, whereas if you look at Amazon – “Oh yeah, it used to do this and it doesn’t do that anymore,” or they took it away and they replaced it with something else that worked better. So there’s that whole real-time instrumentation aspect. Shadow apps, figuring out what people are actually doing which is another way, of course, that users are involved as co-creators of the software.

But users, I mean, for businesses, too. Like for a user to use Yahoo or businesses to get software as a service, they’re not buying a stand-alone old software application. And that’s kind of – the genie’s out of the bottle—that’s known. They’re buying a service where, like you said, the application developers has to be imbedded in that relationship. And it changes the software developers’ focus.

And you got user behavior and the collective intelligence, and you talk about data. Talk about the data and how the Internet now, ticking software development real time, businesses trying to do work and trying to either sell something and serve their clients and users. How does that thread in with these new network effects?

Well first of all, I guess I would just say there are businesses that are not data businesses, obviously. Maybe it’s true that in the future all businesses will be data businesses. I think – I’m making and selling shoes, you know? Well, suppose as we move in the future towards custom manufacturing, knowing how many people like a particular kind of shoe, what kinds of things that they wanna do to customize it, getting the users invested in building their own shoes—you know, all those things are sort of data businesses.

I love to point people to Threadless.com, which is for T-shirts. Here’s a custom T-shirt business that has sold out of every product they’ve ever produced. Why? Because they built a community that proposes new products, that votes on new products; and when there’s sufficient demand, the company says, “Oh okay, that’s a good one.” They manufacture it and sell it to all the people who voted it up—said, “I’d buy that one.” And I think that says a lot for the future of manufacturing.

That’s something we are really looking at hard at O’Reilly—could we do that for books? You know, do a real-time auction before we start the project. There’s actually a fascinating company called, “Logos Bible Software” that actually does electronic editions of various religious and pretty deep scholarly works. “Okay, here’s this Aramaic dictionary that was published in 1900, and it’s out of print, but should we do an electronic edition?” They get people to say, “Yeah, I’d buy it. Here’s how much I’d pay.” They set the price interactively with their audience. They have a little dashboard that says, “There aren’t enough people who said they’d buy it yet for us to do it.” It goes over the threshold, they go, “Okay, we’re in production because enough people said they’d pay us this much money.” Their users are telling them what to make.

Where is the direction of collaborative filtering going today in terms of a technical solution? Where's the direction of that?

I guess what I would say about collaborative filtering—it's one of many, many techniques for doing what I would call harnessing collective intelligence. At the end of the day, another way of telling the Web 2.0 story is that these are all applications for collective intelligence, whether it's an explicit application like a future's market where people are placing bets on some particular outcome, or explicit like reviews and ratings. Or more powerfully, implicit data, because the fact is we're all leaving tracks in cyberspace. There are two kinds of collaborative filtering. There's the kind where people will do comparative ratings. They'll say, "I like this movie, I like this movie, I like this movie," and then the software compares you to all the other people who are like you in their preferences and then makes a recommendation. But there's also a kind of implicit collaborative filtering which is based not on you actually telling people you like something, but on what you actually do. So if you look at the way Amazon does it, they basically say, "People who bought this also bought..." They say, "People who looked at this page tended to buy..." So they're actually gathering data from their users based on implicit behavior. So I think you have to keep that distinction in mind, that there's both implicit and explicit sources of data for collaborative intelligence applications.

And then it's really a matter of can you apply clever algorithms to extract meaning from that data? And I think that going back to the core ideas of Web 2.0, it's really about meaning. And of course, meaning is one of the key ideas of the semantic web. But where I distinguish Web 2.0 and the semantic web is that the semantic web proponents think of meaning as something that needs to be encoded or added to the web. That is, we'll use some kind of mark-up to indicate the meaning in ways that computers can recognize. And many of the breakthroughs in Web 2.0 have come from is the realization that the meaning is already there.

So when Google did page rank, they went, "Oh wait a minute. There's this additional layer of meaning in links. They're actually a vote. And if we can step back enough and we can look at the link structure of the web and we can think about that as a reputation system, that's really meaningful."

Let's talk about that—page rank. And that's a great algorithm of that data that was available at the time, which they created and created relevance and meaning from. Today Search is basically inherently built on that kind of Google paradigm. Type in some key words contextually, get a list of organic search results and some ads on the side and whatnot. But you mentioned that these implicit and explicit things that are coming out, or data, there's algorithms of that data. What is the next page rank in your mind? What do you see in the search world and/or data world collectively—that mixed algorithm?

Well I think you're mixing up two things. I mean, page rank is an algorithm, but it's also a particular data set against which the algorithm is applied. So there are two angles on that. One is the realization of meaning in some additional data sets. So a great example is a company I'm an investor in called "Wesabe." They realize that where people spend their money is a vote, just like making a link is a vote. So if somebody goes to a restaurant once, maybe it doesn't mean anything. If they go back every week, it means a lot. It means more than any rating you can give, right?

So that's one approach - identifying some particular data vector that constitutes a meaning vector. The other is some new algorithm. You say, "Does Bayesian filtering work better than a Hidden Markov model in predicting what people are gonna do next?" And there will be breakthroughs in algorithms, but most of the breakthroughs are going to come from either discovering a data vector that already exists and was just staring us in the face and we didn't realize how much meaning was already encoded there.

Or second, there's a huge opportunity in creating a context in which people encode meaning without thinking that they're doing so. So Facebook is an example of the latter. The whole idea of the social graph was out there for quite a while, and social networking sites were available - Friendster and all that. But the semantic web approach was, "Well, we'll develop a language for expressing relationships so we have both, right?"

Facebook was the first company that really started to say, "Well you know, these consumer apps - and they're about friends and they're not very rich and they're not very powerful in a lot of ways but this is a social graph. And look, let's actually think about it that way and then let's build some tools to let people actually access and exploit that social graph."

So they launched the F8 platform, and the rest is history. So it was the combination of creating a context in which people are actually creating a data source virally that can then be exploited by algorithms. I don't think Facebook has yet had the breakthrough that Google had, 'cause I think Google is really two things. It's a search engine, but it is also a monetization engine.

They developed algorithms for finding better content, but they also developed this really unique way of harnessing the users in a productive way to actually make money, and that was the ad auction. And the AdWords ad auction, again, is this amazing insight into how to harness user contribution. And I can't emphasize how much the breakthrough comes when you say, "What can we do that really values the user?"

If you don't value the user, you say, "Wow, let's charge - let's sell the top ad to the highest bidder." Then the user is just raw material for you, right? But when you value the user as Google did originally, they said, "We wanna give the user the best results." So they started measuring the click-through rate on ads, and then they had this brilliant breakthrough. A lot of people would say, "Oh yeah, well Google AdWords model, it just came from Overture." It's not true. Yahoo bought Overture and they didn't switch till nearly five years after Google to this ad auction model in which the real-time measurement of what the click-through rate would be trumped price.

Google realized that user experience dictated that they liked it

That's right. But it also made more money. If five times as many people click on the ad, you're better off selling an ad that only has half the price. You still get twice as much money, right? Or two and a half times as much money. So there's this wonderful thing that happened where it becomes a virtuous circle. So I think the breakthroughs will come when we find ways that really serve the user.

Let's take Web 2.0 and push it out to the kind of the Joe six-pack businessman and retailers out there. Aggregate Knowledge is sponsored here, but they sell to retailers their technology. And media companies out there are trying to be like Google or like Web 2.0-like approaches. What key technologies should these online retailers and media companies embark on to be successful? I mean, the user is one thing. They say they "are in favor of their customer," but what should they be looking at from a technology perspective?

I think one thing that we definitely see in the Internet Operating System – and again, we'll come back to that term—is that this is a set of system services. You don't have to build everything yourself. And I think IT departments are used to thinking in terms of the space that they control. Inside their firewall, there's an application space. And they would say, "Well, I would never use these services from some outside software or service vendor," you know? And I go, "Wait a minute. You buy an operating system like Windows from an outside vendor and you're happy to call their services to access your printers and your disks and your screens," right? And they forget that back in the eighties, every manufacturer of computer equipment was writing their own device drivers. And Microsoft came along and said, "No, no, we'll take care of that."

And we haven't quite reached that tipping point yet, but what you need to think about are there are effectively the equivalent of device drivers for various kinds of data services out there on the web. And when you're building an application, you want to build the unique parts that only you can build and you want to re-use the services from outside companies where that makes sense.

Think about it just like, "Hey, I'm buying a disk; I'm getting a device driver." You know? "I'm buying – I want to use collaborative filtering, I'm going to get this service from here. I want to use storage, I'm going to use this feature from Amazon. I want to get identity, in the future I'll be getting it from Facebook. Or maybe I'll get it from Google Open Social or whatever. If I'm going to do a search, maybe I'll use Google" Again, there's going to be a rich ecosystem. But you can think of all of these things right now that look like applications and start thinking of them like subsystems of the new operating system.

So doing siloed-based solutions really talks about the old model. And what you're saying is for a retailer or media company to let go a little bit and put a piece of JavaScript on their website, leverage a service or a product or application that someone else built?

Absolutely. And also realize where you're already doing it. They were starting – I mean, when people started using DoubleClick. It's a little bit of code hidden on your web page and it's literally pulling an ad banner from some other random place on the web and making it part of your user experience.

So it's been happening for a long time; it was just people didn't realize it was happening. What happens as a platform evolves is people realize more and more deeply, "Oh, that's what's going on."

There have been web services around for quite a while, but they weren't the norm. So I think of DoubleClick—one of the first web services. Another one: CDDDB. You stick a CD in and you go, "Wow, magic! It's looking up the track names." Well, that was a web service, right?

The Internet Operating System

Tim O'Reilly

Discovery
Expert
Series

Just a final topic and we can close this out, Tim, is the Internet Operating System. It's been talked about and it's been re-purposed in terms of description, but we are living in an operating system environment with the services. So talk about your vision, some research that you uncovered in this area, and what you see as the forward-looking trend there.

Well the biggest thing I would like to close with is that we have a choice of the kind of operating system we want. If you look at the history of the last 20 years - 25 years maybe - there have really been two dominant operating system traditions. And one is represented well by the Microsoft operating systems and that is a single vendor offers the operating system. They control the features.

Microsoft has done a great job of providing a huge amount of value. They built this PC industry—this PC ecosystem.

The other model is the one that's represented by UNIX, by Linux, and the Internet. And if you think about those alternate operating systems, you realize that no one company created or owns all the pieces. A company like Red Hat that assembles a Linux distribution is really dealing with a software supply chain that it does not control. There's six or seven thousand programs assembled there. There's a set of rules by which they inter-operate. The same thing with Internet software that came on stream, you know? You got your domain name system from Paul Vixie and the Internet Software Consortium. You got your web browser from, originally, Mosaic and then maybe from Microsoft and Mozilla. And you can also have one from Apple. It's standards-based. It's open systems. It's interoperability.

And right now, we're in this world of these new web services coming on stream. And what's really great is they're coming from multiple vendors. But we are in a place where, there are vendors who would love to own the entire space and say, "Be on our platform." And it's really important for companies to understand that's a dangerous choice. Back when Microsoft, in the eighties, said to application developers, "Hey, don't worry about writing all those device drivers. We'll take care of it for you," that led to an era of lock-in.

It's really important for us to have a multi-vendor world—a world where we get the best of breed services from a lot of players, and that those services are interoperable. So one of the principles I would just really like to leave people with is make sure that you don't get locked in. Ask for interoperability.

Are we in an environment where just lock-in is now fantasy? Has it become so prevalent with the Internet in this distributed architecture that, really, no one can own it?

No, not at all. This goes back to the very beginnings and the heart of my activism about Web 2.0, starting with my arguments about open source. I believe that the natural tendency of the Internet with network effects is towards concentration of power. If you look at a Google, for example they're getting more and more powerful. There are not other companies sort of gaining on them in search.

So yes, the open standards aspect allows for innovation to happen, but once innovation happens, companies gain momentum. And I think more and more, we'll end up with a best of breed in each category, and those guys are going to start buying each other. And there may come a time when a single company owns enough of the key features

that they will then try to force people to use their other components even though somebody else's are better.

Is that avoidable or unavoidable in terms of the net – one of the benefits of the net effects and, even as you talk about this data which there are new data sets emerging and algorithms that there creates a monopolistic framework? I mean, Google, for example, is at it, but in today's Web 2.0 world, free distribution for media, social networks emerging, open social kind of out there as a stalking horse for this kind of medium ground.

Why is Google doing Open Social? Why? Because Facebook was really gaining traction as the social network operating system because they thought like a platform player. MySpace was just an application, but Facebook said, "We understand what it means to be a platform. We're going to be the social network platform." We're actually just entering into the phase of Web 2.0 that I would call "the platform wars," where people are trying to get competitive advantage; where Microsoft says, "Google is getting so much advantage in search that we have to buy Yahoo so we can have an effective counterweight because they're going to take it all if we don't." And they may take it all anyway.

Amazon is getting a really interesting foothold in becoming an infrastructure level player for storage and compute services, and that has some really interesting potential. I think we're going to see various areas break out. And then again, I think they'll start to aggregate. People will buy one another. So I think that what we have to do is make sure that as that happens that there is this effective counterweight.

So for the folks out there, just to summarize and close it out here, you got your business guy out there, you have some developers and corporations, you have freelance software developers and you have users. What do you say to those folks with these platform wars in the Internet operating system? What can they do? How can they help themselves to continue to improve, continuously innovate, and how should they look at this kind of scene that they're approaching, which is platform wars, web services from multiple vendors, network effects. How do they tap into that and how do they leverage that?

Well I think, first of all, there's never been a better time to try things. We have these lightweight development methodologies, easy to try and fail - try and succeed. People can go out there and make something happen on Facebook. There literally was a course that was run at Stanford where the students got their grade by the number of users that they attracted on Facebook. And a couple of applications – just blows it out. Now those guys are actually leaving school and they're starting a company.

So there's still a huge amount of opportunity. I guess I would say, first of all, advice would be don't just think about your current product set; think about your customer and what you really do. What job do you do? So for example, at O'Reilly, we didn't say, "Oh yeah, we're a book publisher." We said, "We're a company that's about innovation. In fact, our company mission is, "Changing the World by Spreading the Knowledge of Innovators." So we went from books into online publishing and into conferences and early-stage investing and online subscription services. Because we said, "Wow, all those things are ways that we do that job, which is to find people who are doing new and interesting technology and then help add information to help other people follow in their footsteps."

So I guess that would be the first and most important piece of advice. Remember that

The Internet Operating System

Tim O'Reilly

Discovery
Expert
Series

what you do isn't defined by just the products you offer today. It's the service that you offer for your customers. And that service can be expressed in many, many different kinds of products, in many different ways. And there's this fabulous arsenal of tools that you can use so that you don't have to do it all yourself to build an online equivalent to those services. And so just get out there and be creative.

Well there it is, Tim O'Reilly talking about what it means for businesses and people building products and technologies in the Web 2.0 and modern web, semantic web, Internet operating system. A lot of very thought-provoking and great directions on some of those new technologies. We're here with Tim O'Reilly from the Discovery Expert series, sponsored by Aggregate Knowledge. Tim, thank you very much.

aggregate
knowledge

1510 Fashion Island Blvd., Suite 201
San Mateo, CA 94404

Tel 650.293.2920 | Fax 650.293.2919

www.aggregateknowledge.com

Copyright ©2007 Aggregate Knowledge, Inc. All rights reserved. Aggregate Knowledge, the Aggregate Knowledge logo, Pique, the Pique logo, Pique Discovery, Pique Onsite, Pique Email, Pique Affiliate, Pique Multi-Site, Pique Discovery Window, and Pique Discovery Igniter are trademarks of Aggregate Knowledge Corporation in the United States, other countries, or both. All other company, product, or service names may be trademarks or service marks of their respective owners. All statements regarding plans, directions, and intent of Aggregate Knowledge are subject to change without notice. **DESB-1107-01**