4

What Is Web 2.0?

Design Patterns and Business Models for the Next Generation of Software

TIM O'REILLY

The bursting of the dot-com bubble in the fall of 2001 marked a turning point for the web. Many people concluded that the web was overhyped, when in fact bubbles and consequent shakeouts appear to be a common feature of all technological revolutions.[1] Shakeouts typically mark the point at which an ascendant technology is ready to take its place at center stage. The pretenders are given the bum's rush, the real success stories show their strength, and there begins to be an understanding of what separates one from the other.

The concept of "Web 2.0" began with a conference brainstorming session between O'Reilly Media and MediaLive International. Dale Dougherty, web pioneer and O'Reilly vice president, noted that far from having "crashed," the web was more important than ever, with exciting new applications and sites popping up with surprising regularity. What's more, the companies that had survived the collapse seemed to have some things in common. Could it be that the dot-com collapse marked some kind of turning point for the web, such that a call to action such as "Web 2.0" might make sense? We agreed that it did, and so the Web 2.0 Conference was born.[2]

In the year and a half since, the term "Web 2.0" has clearly taken hold, with more than 9.5 million citations in Google. But there's still a huge amount of disagreement about just what "Web 2.0" means,[3] with some people decrying it as a meaningless marketing buzzword and others accepting it as the new conventional wisdom.

This essay is an attempt to clarify just what we mean by "Web 2.0." In our initial brainstorming, we formulated our sense of Web 2.0 by example (see table 4.1). The list went on and on. But what was it that made us identify one application or approach as "Web 1.0" and another as "Web 2.0"? (The question is particularly urgent because the Web 2.0 meme has become so

TABLE 4.1

Web 1.0	*Web 2.0*
DoubleClick	Google AdSense
Ofoto	Flickr
Akamai	BitTorrent
mp3.com	Napster
Britannica Online	Wikipedia
personal websites	blogging
evite	upcoming.org and EVDB
domain name speculation	search engine optimization
page views	cost per click
screen scraping	web services
publishing	participation
content management systems	wikis
directories (taxonomy)	tagging ("folksonomy")
stickiness	syndication

widespread that companies are now pasting it on as a marketing buzzword, with no real understanding of just what it means. The question is particularly difficult because many of those buzzword-addicted start-ups are definitely *not* Web 2.0, while some of the applications we identified as Web 2.0, like Napster and BitTorrent, are not even properly web applications!) We began trying to tease out the principles that are demonstrated in one way or another by the success stories of Web 1.0 and by the most interesting of the new applications.

1. The Web as Platform

Like many important concepts, Web 2.0 doesn't have a hard boundary but, rather, a gravitational core. You can visualize Web 2.0 as a set of principles and practices that tie together a veritable solar system of sites that demonstrate some or all of those principles, at a varying distance from that core. Figure 4.1 shows a "meme map" of Web 2.0 that was developed at a brainstorming session during FOO Camp, a conference at O'Reilly Media. It's very much a work in progress, but it shows the many ideas that radiate out from the Web 2.0 core.

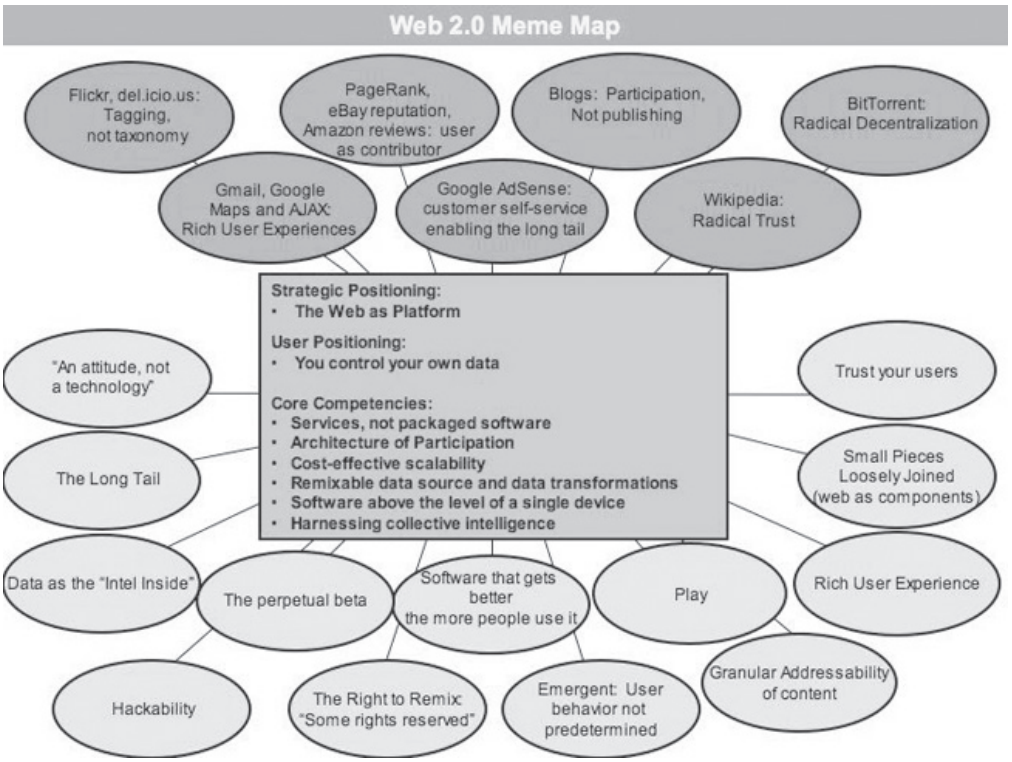

Fig. 4.1. Web 2.0 meme map

For example, at the first Web 2.0 conference, in October 2004, John Battelle and I listed a preliminary set of principles in our opening talk. The first of those principles was "the web as platform." Yet that was also a rallying cry of Web 1.0 darling Netscape, which went down in flames after a heated battle with Microsoft. What's more, two of our initial Web 1.0 exemplars, DoubleClick and Akamai, were both pioneers in treating the web as a platform. People don't often think of ad serving as "web services," but in fact, it was the first widely deployed web service and the first widely deployed "mashup" (to use another term that has gained currency of late). Every banner ad is served as a seamless cooperation between two websites, delivering an integrated page to a reader on yet another computer. Akamai also treats the network as the platform, and at a deeper level of the stack, building a transparent caching and content-delivery network that eases bandwidth congestion.

Nonetheless, these pioneers provided useful contrasts because later entrants have taken their solution to the same problem even further, understanding something deeper about the nature of the new platform. Both DoubleClick and Akamai were Web 2.0 pioneers, yet we can also see how it's possible to realize more of the possibilities by embracing additional Web 2.0 design patterns.

Let's drill down for a moment into each of these three cases, teasing out some of the essential elements of difference.

Netscape vs. Google

If Netscape was the standard bearer for Web 1.0, Google is most certainly the standard bearer for Web 2.0, if only because their respective initial public offerings were defining events for each era. So let's start with a comparison of these two companies and their positioning.

Netscape framed "the web as platform" in terms of the old software paradigm: its flagship product was the web browser, a desktop application, and its strategy was to use its dominance in the browser market to establish a market for high-priced server products. Control over standards for displaying content and applications in the browser would, in theory, give Netscape the kind of market power enjoyed by Microsoft in the PC market. Much like the "horseless carriage" framed the automobile as an extension of the familiar, Netscape promoted a "webtop" to replace the desktop and planned to populate that webtop with information updates and applets pushed to the webtop by information providers who would purchase Netscape servers.

In the end, both web browsers and web servers turned out to be commodities, and value moved "up the stack" to services delivered over the web platform.

Google, by contrast, began its life as a native web application, never sold or packaged but delivered as a service, with customers paying, directly or indirectly, for the use of that service. None of the trappings of the old software industry is present. No scheduled software releases, just continuous improvement. No licensing or sale, just usage. No porting to different platforms so that customers can run the software on their own equipment, just a massively scalable collection of commodity PCs running open-source operating systems plus homegrown applications and utilities that no one outside the company ever gets to see.

At bottom, Google requires a competency that Netscape never needed: database management. Google isn't just a collection of software tools; it's a specialized database. Without the data, the tools are useless; without the software, the data is unmanageable.

DoubleClick vs. Overture and AdSense

Like Google, DoubleClick is a true child of the Internet era. It harnesses software as a service, has a core competency in data management, and, as

noted earlier, was a pioneer in web services long before web services even had a name. However, DoubleClick was ultimately limited by its business model. It bought into the '90s notion that the web was about publishing, not participation; that advertisers, not consumers, ought to call the shots; that size mattered; and that the Internet was increasingly being dominated by the top websites as measured by MediaMetrix and other web ad-scoring companies.

As a result, DoubleClick proudly cites on its website "over 2000 successful implementations" of its software. Yahoo! Search Marketing (formerly Overture) and Google AdSense, by contrast, already serve hundreds of thousands of advertisers apiece.

Overture's and Google's success came from an understanding of what Chris Anderson refers to as "the long tail," the collective power of the small sites that make up the bulk of the web's content (see chapter 10 in this volume). DoubleClick's offerings require a formal sales contract, limiting its market to the few thousand largest websites. Overture and Google figured out how to enable ad placement on virtually any web page. What's more, they eschewed publisher/ad-agency-friendly advertising formats such as banner ads and popups in favor of minimally intrusive, context-sensitive, consumer-friendly text advertising.

The Web 2.0 lesson: *leverage customer self-service and algorithmic data management to reach out to the entire web, to the edges and not just the center, to the long tail and not just the head*.

Not surprisingly, other Web 2.0 success stories demonstrate this same behavior. eBay enables occasional transactions of only a few dollars between single individuals, acting as an automated intermediary. Napster (though shut down for legal reasons) built its network not by building a centralized song database but by architecting a system in such a way that every downloader also became a server and thus grew the network.

Akamai vs. BitTorrent

Like DoubleClick, Akamai is optimized to do business with the head, not the tail, with the center, not the edges. While it serves the benefit of the individuals at the edge of the web by smoothing their access to the high-demand sites at the center, it collects its revenue from those central sites.

BitTorrent, like other pioneers in the peer-to-peer (P2P) movement, takes a radical approach to Internet decentralization. Every client is also a server; files are broken up into fragments that can be served from multiple locations,

transparently harnessing the network of downloaders to provide both bandwidth and data to other users. The more popular the file, in fact, the faster it can be served, as there are more users providing bandwidth and fragments of the complete file.

BitTorrent thus demonstrates a key Web 2.0 principle: *the service automatically gets better the more people use it*. While Akamai must add servers to improve service, every BitTorrent consumer brings his or her own resources to the party. There's an implicit "architecture of participation," a built-in ethic of cooperation, in which the service acts primarily as an intelligent broker, connecting the edges to each other and harnessing the power of the users themselves.

2. Harnessing Collective Intelligence

The central principle behind the success of the giants born in the Web 1.0 era who have survived to lead the Web 2.0 era appears to be this, that they have embraced the power of the web to harness collective intelligence:

Hyperlinking is the foundation of the web. As users add new content, and new sites, it is bound in to the structure of the web by other users discovering the content and linking to it. Much as synapses form in the brain, with associations becoming stronger through repetition or intensity, the web of connections grows organically as an output of the collective activity of all web users.

Yahoo!, the first great Internet success story, was born as a catalog, or directory of links, an aggregation of the best work of thousands and then millions of web users. While Yahoo! has since moved into the business of creating many types of content, its role as a portal to the collective work of the net's users remains the core of its value.

Google's breakthrough in search, which quickly made it the undisputed search-market leader, was PageRank, a method of using the link structure of the web rather than just the characteristics of documents to provide better search results.

eBay's product is the collective activity of all its users; like the web itself, eBay grows organically in response to user activity, and the company's role is as an enabler of a context in which that user activity can happen. What's more, eBay's competitive advantage comes almost entirely from the critical mass of buyers and sellers, which makes any new entrant offering similar services significantly less attractive.

Amazon sells the same products as competitors such as Barnesandnoble.com, and it receives the same product descriptions, cover images, and editorial content from its vendors. But Amazon has made a science of user engagement. It has an order of magnitude more user reviews, invitations to participate in varied ways on virtually every page—and, even more important, it uses user activity to produce better search results. While a Barnesandnoble.com search is likely to lead with the company's own products, or sponsored results, Amazon always leads with "most popular," a real-time computation based not only on sales but on other factors that Amazon insiders call the "flow" around products. With an order of magnitude more user participation, it's no surprise that Amazon's sales also outpace competitors'.

Now, innovative companies that pick up on this insight and perhaps extend it even further are making their mark on the web:

Wikipedia, an online encyclopedia based on the unlikely notion that an entry can be added by any web user and edited by any other, is a radical experiment in trust, applying Eric Raymond's dictum (originally coined in the context of open-source software) that "with enough eyeballs, all bugs are shallow" to content creation.[4] Wikipedia is already in the top one hundred websites, and many people think it will be in the top ten before long. This is a profound change in the dynamics of content creation!

Sites like del.icio.us and Flickr, two companies that have received a great deal of attention of late, have pioneered a concept that some people call "folksonomy" (in contrast to taxonomy), a style of collaborative categorization of sites using freely chosen keywords, often referred to as tags.[5] Tagging allows for the kind of multiple, overlapping associations that the brain itself uses, rather than rigid categories. In the canonical example, a Flickr photo of a puppy might be tagged both "puppy" and "cute"—allowing for retrieval along natural axes-generated user activity.

Collaborative spam-filtering products like Cloudmark aggregate the individual decisions of e-mail users about what is and is not spam, outperforming systems that rely on analysis of the messages themselves.

It is a truism that the greatest Internet success stories don't advertise their products. Their adoption is driven by "viral marketing"—that is, recommendations propagating directly from one user to another. You can almost make the case that if a site or product relies on advertising to get the word out, it isn't Web 2.0.

Even much of the infrastructure of the web—including the Linux, Apache, MySQL, and Perl, PHP, or Python code involved in most web servers—relies on the peer-production methods of open source, in themselves an instance of collective, net-enabled intelligence.[6] There are more than one hundred thousand open-source software projects listed on SourceForge.net. Anyone can add a project, anyone can download and use the code, and new projects migrate from the edges to the center as a result of users putting them to work, an organic software-adoption process relying almost entirely on viral marketing.

The lesson: *network effects from user contributions are the key to market dominance in the Web 2.0 era*.

Software licensing and control over application programming interfaces (APIs)—the lever of power in the previous era—is irrelevant because the software never need be distributed but only performed, and also because without the ability to collect and manage the data, the software is of little use. In fact, *the value of the software is proportional to the scale and dynamism of the data it helps to manage*.

Google's service is not a server, though it is delivered by a massive collection of Internet servers; nor is it a browser, though it is experienced by the user within the browser. Nor does its flagship search service even host the content that it enables users to find. Much like a phone call, which happens not just on the phones at either end of the call but on the network in between, Google happens in the space between browser and search engine and destination content server, as an enabler or middleman between the user and his or her online experience.

While both Netscape and Google could be described as software companies, it's clear that Netscape belonged to the same software world as Lotus, Microsoft, Oracle, SAP, and other companies that got their start in the 1980s software revolution, while Google's fellows are other Internet applications like eBay, Amazon, Napster, and, yes, DoubleClick and Akamai.

Blogging and the Wisdom of Crowds

One of the most highly touted features of the Web 2.0 era is the rise of blogging. Personal home pages have been around since the early days of the web, and the personal diary and daily opinion column have been around much longer than that. So just what is the fuss all about?

A blog, at its most basic, is just a personal home page in diary format. But as Rich Skrenta notes, the chronological organization of a blog "seems like a trivial difference, but it drives an entirely different delivery, advertising and value chain."[7]

One of the things that has made a difference is a technology called RSS.[8] RSS is the most significant advance in the fundamental architecture of the web since early hackers realized that CGI could be used to create database-backed websites. RSS allows someone not just to link to a page but to subscribe to it, with notification every time that page changes. Skrenta calls this "the incremental web." Others call it the "live web."

Now, of course, "dynamic websites" (i.e., database-backed sites with dynamically generated content) replaced static web pages well over ten years ago. What's dynamic about the live web are not just the pages but the links. A link to a weblog is expected to point to a perennially changing page, with "permalinks" for any individual entry and notification for each change. An RSS feed is thus a much stronger link than, say, a bookmark or a link to a single page.

RSS also means that the web browser is not the only means of viewing a web page. While some RSS aggregators, such as Bloglines, are web based, others are desktop clients, and still others allow users of portable devices to subscribe to constantly updated content.

RSS is now being used to push not just notices of new blog entries but also all kinds of data updates, including stock quotes, weather data, and photo availability. This use is actually a return to one of its roots: RSS was born in 1997 out of the confluence of Dave Winer's "Really Simple Syndication" technology, used to push out blog updates, and Netscape's "Rich Site Summary," which allowed users to create custom Netscape home pages with regularly updated data flows. Netscape lost interest, and the technology was carried forward by blogging pioneer Userland, Winer's company. In the current crop of applications, though, we see the heritage of both parents.

But RSS is only part of what makes a weblog different from an ordinary web page. Tom Coates remarks on the significance of the permalink:

> It may seem like a trivial piece of functionality now, but it was effectively the device that turned weblogs from an ease-of-publishing phenomenon into a conversational mess of overlapping communities. For the first time it became relatively easy to gesture directly at a highly specific post on someone else's site and talk about it. Discussion emerged. Chat emerged. And—as a result—friendships emerged or became more entrenched. The permalink was the first—and most successful—attempt to build bridges between weblogs.[9]

In many ways, the combination of RSS and permalinks adds many of the features of NNTP, the Network News Protocol of the Usenet, onto HTTP, the web protocol. The "blogosphere" can be thought of as a new, peer-to-peer equivalent to Usenet and bulletin boards, the conversational watering holes of the early Internet. Not only can people subscribe to each other's sites and easily link to individual comments on a page, but also, via a mechanism known as trackbacks, they can see when anyone else links to their pages and can respond, either with reciprocal links or by adding comments.

Interestingly, two-way links were the goal of early hypertext systems like Xanadu. Hypertext purists have celebrated trackbacks as a step toward two-way links. But note that trackbacks are not properly two way—rather, they are really (potentially) symmetrical one-way links that create the effect of two-way links. The difference may seem subtle, but in practice it is enormous. Social networking systems like Friendster, Orkut, and LinkedIn, which require acknowledgment by the recipient in order to establish a connection, lack the same scalability as the web. As noted by Caterina Fake, cofounder of the Flickr photo-sharing service, attention is only coincidentally reciprocal. (Flickr thus allows users to set watch lists—any user can subscribe to any other user's photostream via RSS. The object of attention is notified but does not have to approve the connection.)

If an essential part of Web 2.0 is harnessing collective intelligence, turning the web into a kind of global brain, the blogosphere is the equivalent of constant mental chatter in the forebrain, the voice we hear in all of our heads. It may not reflect the deep structure of the brain, which is often unconscious, but is instead the equivalent of conscious thought. And as a reflection of conscious thought and attention, the blogosphere has begun to have a powerful effect.

First, because search engines use link structure to help predict useful pages, bloggers, as the most prolific and timely linkers, have a disproportionate role in shaping search-engine results. Second, because the blogging community is so highly self-referential, bloggers' paying attention to other bloggers magnifies their visibility and power. The "echo chamber" that critics decry is also an amplifier.

If blogging were merely an amplifier, it would be uninteresting. But like Wikipedia, blogging harnesses collective intelligence as a kind of filter. What James Surowiecki calls "the wisdom of crowds" comes into play, and much as PageRank produces better results than analysis of any individual document, the collective attention of the blogosphere selects for value.[10]

While mainstream media may see individual blogs as competitors, what is really unnerving is that the competition is with the blogosphere as a whole. This is not just a competition between sites but a competition between business models. The world of Web 2.0 is also the world of what Dan Gillmor calls "we, the media," a world in which "the former audience," not a few people in a back room, decides what's important.[11]

3. Data Is the Next Intel Inside

Every significant Internet application to date has been backed by a specialized database: Google's web crawl, Yahoo!'s directory (and web crawl), Amazon's database of products, eBay's database of products and sellers, MapQuest's map databases, Napster's distributed song database. As Hal Varian remarked in a personal conversation last year, "SQL is the new HTML." Database management is a core competency of Web 2.0 companies, so much so that we have sometimes referred to these applications as "infoware" rather than merely software.[12]

This fact leads to a key question: Who owns the data?

In the Internet era, one can already see a number of cases where control over the database has led to market control and outsized financial returns. The monopoly on domain-name registry initially granted by government fiat to Network Solutions (later purchased by Verisign) was one of the first great moneymakers of the Internet. While we've argued that business advantage via controlling software APIs is much more difficult in the age of the Internet, control of key data sources is not, especially if those data sources are expensive to create or amenable to increasing returns via network effects.

Look at the copyright notices at the base of every map served by MapQuest, maps.yahoo.com, maps.msn.com, or maps.google.com, and you'll see the line "Maps copyright NavTeq, TeleAtlas" or, with the new satellite-imagery services, "Images copyright Digital Globe." These companies made substantial investments in their databases (NavTeq alone reportedly invested $750 million to build its database of street addresses and directions. Digital Globe spent $500 million to launch its own satellite to improve on government-supplied imagery.) NavTeq has gone so far as to imitate Intel's familiar "Intel Inside" logo: cars with navigation systems bear the imprint "NavTeq Onboard." Data is indeed the "Intel Inside" of these applications, a sole source component in systems whose software infrastructure is largely open source or otherwise commodified.

The now hotly contested web-mapping arena demonstrates how a failure to understand the importance of owning an application's core data will eventually undercut its competitive position. MapQuest pioneered the web-mapping category in 1995, yet when Yahoo! and then Microsoft and most recently Google decided to enter the market, they were easily able to offer a competing application simply by licensing the same data.

Contrast, however, the position of Amazon.com. Like competitors such as Barnesandnoble.com, its original database came from ISBN registry provider R. R. Bowker. But unlike MapQuest, Amazon relentlessly enhanced the data, adding publisher-supplied data such as a cover image, a table of contents, an index, and sample material. Even more importantly, it harnessed its users to annotate the data, such that after ten years, Amazon, not Bowker, is the primary source for bibliographic data on books, a reference source for scholars and librarians as well as consumers. Amazon also introduced its own proprietary identifier, the ASIN, which corresponds to the ISBN when one is present and creates an equivalent name space for products without one. Effectively, Amazon "embraced and extended" its data suppliers.

Imagine if MapQuest had done the same thing, harnessing its users to annotate maps and directions, adding layers of value. It would have been much more difficult for competitors to enter the market just by licensing the base data.

The recent introduction of Google Maps provides a living laboratory for the competition between application vendors and their data suppliers. Google's lightweight programming model has led to the creation of numerous value-added services in the form of mashups that link Google Maps with other Internet-accessible data sources. Paul Rademacher's housingmaps.com, which combines Google Maps with Craigslist apartment-rental and home-purchase data to create an interactive housing search tool, is the preeminent example of such a mashup.

At present, these mashups are mostly innovative experiments, done by hackers. But entrepreneurial activity follows close behind. And already one can see that for at least one class of developer, Google has taken the role of data source away from NavTeq and inserted itself as a favored intermediary. We expect to see battles between data suppliers and application vendors in the next few years, as both realize just how important certain classes of data will become as building blocks for Web 2.0 applications.

The race is on to own certain classes of core data: location, identity, calendaring of public events, product identifiers, and name spaces. In many cases, where there is significant cost to create the data, there may be an opportunity for an Intel Inside–style play, with a single source for the data. In others, the

winner will be the company that first reaches critical mass via user aggregation and turns that aggregated data into a system service.

For example, in the area of identity, PayPal, Amazon's 1-click, and the millions of users of communications systems may all be legitimate contenders to build a network-wide identity database. (In this regard, Google's recent attempt to use cell-phone numbers as an identifier for Gmail accounts may be a step toward embracing and extending the phone system.) Meanwhile, start-ups like Sxip are exploring the potential of federated identity, in quest of a kind of "distributed 1-click" that will provide a seamless Web 2.0 identity subsystem. In the area of calendaring, EVDB is an attempt to build the world's largest shared calendar via a wiki-style architecture of participation. While the jury's still out on the success of any particular start-up or approach, it's clear that standards and solutions in these areas, effectively turning certain classes of data into reliable subsystems of the "Internet operating system," will enable the next generation of applications.

A further point must be noted with regard to data, and that is user concerns about privacy and their rights to their own data. In many of the early web applications, copyright is only loosely enforced. For example, Amazon lays claim to any reviews submitted to the site, but in the absence of enforcement, people may repost the same review elsewhere. However, as companies begin to realize that control over data may be their chief source of competitive advantage, we may see heightened attempts at control.

Much as the rise of proprietary software led to the Free Software movement, we expect the rise of proprietary databases to result in a Free Data movement within the next decade. One can see early signs of this countervailing trend in open data projects such as Wikipedia, in the Creative Commons, and in software projects like Greasemonkey, which allow users to take control of how data is displayed on their computer.

4. End of the Software Release Cycle

As noted earlier in the discussion of Google versus Netscape, one of the defining characteristics of Internet-era software is that it is delivered as a service, not as a product. This fact leads to a number of fundamental changes in the business model of such a company:

Operations must become a core competency. Google's or Yahoo!'s expertise in product development must be matched by an expertise in daily operations. So fundamental is the shift from software as artifact to software as service that *the*

software will cease to perform unless it is maintained on a daily basis. Google must continuously crawl the web and update its indices, continuously filter out link spam and other attempts to influence its results, and continuously and dynamically respond to hundreds of millions of asynchronous user queries, simultaneously matching them with context-appropriate advertisements.

It's no accident that Google's system administration, networking, and load-balancing techniques are perhaps even more closely guarded secrets than are their search algorithms. Google's success at automating these processes is a key part of its cost advantage over competitors.

It's also no accident that scripting languages such as Perl, Python, PHP, and now Ruby play such a large role at Web 2.0 companies. Perl was famously described by Hassan Schroeder, Sun's first webmaster, as "the duct tape of the Internet." Dynamic languages (often called scripting languages and looked down on by the software engineers of the era of software artifacts) are the tool of choice for system and network administrators, as well as for application developers building dynamic systems that require constant change.

Users must be treated as codevelopers, in a reflection of open-source development practices (even if the software in question is unlikely to be released under an open-source license). The open-source dictum, "release early and release often," in fact has morphed into an even more radical position, "the perpetual beta," in which the product is developed in the open, with new features slipstreamed in on a monthly, weekly, or even daily basis. It's no accident that services such as Gmail, Google Maps, Flickr, del.icio.us, and the like may be expected to bear a "Beta" logo for years at a time.

Real-time monitoring of user behavior to see just which new features are used, and how they are used, thus becomes another required core competency. A web developer at a major online service remarked, "We put up two or three new features on some part of the site every day, and if users don't adopt them, we take them down. If they like them, we roll them out to the entire site."

Cal Henderson, the lead developer of Flickr, recently revealed that the company deploys new builds up to every half hour.[13] This is clearly a radically different development model! While not all web applications are developed in as extreme a style as Flickr, almost all web applications have a development cycle that is radically unlike anything from the PC or client-server era. It is for this reason that a recent ZDNet editorial concluded that Microsoft won't be able to beat Google: "Microsoft's business model depends on everyone upgrading their computing environment every two to three years. Google's depends on everyone exploring what's new in their computing environment every day."[14]

While Microsoft has demonstrated enormous ability to learn from and ultimately best its competition, there's no question that this time, the competition will require Microsoft (and by extension, every other existing software company) to become a deeply different kind of company. Native Web 2.0 companies enjoy a natural advantage, as they don't have old patterns (and corresponding business models and revenue sources) to shed.

5. Lightweight Programming Models

Once the idea of web services became au courant, large companies jumped into the fray with a complex web-services stack designed to create highly reliable programming environments for distributed applications.

But much as the web succeeded precisely because it overthrew much of hypertext theory, substituting a simple pragmatism for ideal design, RSS has become perhaps the single most widely deployed web service because of its simplicity, while the complex corporate web-services stacks have yet to achieve wide deployment.

Similarly, Amazon.com's web services are provided in two forms: one adhering to the formalisms of the SOAP (Simple Object Access Protocol) web-services stack, the other simply providing XML data over HTTP, in a lightweight approach sometimes referred to as REST (Representational State Transfer). While high-value business-to-business (B2B) connections (like those between Amazon and retail partners like Toys "R" Us) use the SOAP stack, Amazon reports that 95 percent of the usage is of the lightweight REST service.

This same quest for simplicity can be seen in other "organic" web services. Google's recent release of Google Maps is a case in point. Google Maps' simple AJAX (Javascript and XML) interface was quickly decrypted by hackers, who then proceeded to remix the data into new services.

Mapping-related web services had been available for some time from GIS vendors such as ESRI as well as from MapQuest and Microsoft MapPoint. But Google Maps set the world on fire because of its simplicity. While experimenting with any of the formal vendor-supported web services required a formal contract between the parties, the way Google Maps was implemented left the data for the taking, and hackers soon found ways to creatively reuse that data.

There are several significant lessons here:

Support lightweight programming models that allow for loosely coupled systems. The complexity of the corporate-sponsored web-services stack is designed to enable tight coupling. While this is necessary in many cases, many of the most interesting applications can indeed remain loosely coupled and even fragile. The Web 2.0 mind-set is very different from the traditional IT mind-set!

Think syndication, not coordination. Simple web services, like RSS and REST-based web services, are about syndicating data outward, not controlling what happens when it gets to the other end of the connection. This idea is fundamental to the Internet itself, a reflection of what is known as the end-to-end principle.[15]

Design for "hackability" and remixability. Systems like the original web, RSS, and AJAX all have this in common: the barriers to reuse are extremely low. Much of the useful software is actually open source, but even when it isn't, there is little in the way of intellectual-property protection. The web browser's "View Source" option made it possible for any user to copy any other user's web page; RSS was designed to empower the user to view the content he or she wants, when it's wanted, not at the behest of the information provider; the most successful web services are those that have been easiest to take in new directions unimagined by their creators. The phrase "some rights reserved," which was popularized by the Creative Commons to contrast with the more typical "all rights reserved," is a useful guidepost.

Innovation in Assembly

Lightweight business models are a natural concomitant of lightweight programming and lightweight connections. The Web 2.0 mind-set is good at reuse. A new service like housingmaps.com was built simply by snapping together two existing services. Housingmaps.com doesn't have a business model (yet)—but for many small-scale services, Google AdSense (or perhaps Amazon Associates fees, or both) provides the snap-in equivalent of a revenue model.

These examples provide an insight into another key Web 2.0 principle, which we call "innovation in assembly." When commodity components are abundant, you can create value simply by assembling them in novel or effective ways. Much as the PC revolution provided many opportunities for innovation in assembly of commodity hardware, with companies like Dell making a science out of such assembly, thereby defeating companies whose

business model required innovation in product development, we believe that Web 2.0 will provide opportunities for companies to beat the competition by getting better at harnessing and integrating services provided by others.

6. Software above the Level of a Single Device

One other feature of Web 2.0 that deserves mention is the fact that it's no longer limited to the PC platform. Longtime Microsoft developer Dave Stutz pointed out in his parting advice to Microsoft that "useful software written above the level of the single device will command high margins for a long time to come."[16]

Of course, any web application can be seen as software above the level of a single device. After all, even the simplest web application involves at least two computers: the one hosting the web server and the one hosting the browser. And as we've discussed, the development of the web as platform extends this idea to synthetic applications composed of services provided by multiple computers.

But as with many areas of Web 2.0, where the "2.0-ness" is not something new but rather a fuller realization of the true potential of the web platform, this phrase gives us a key insight into how to design applications and services for the new platform.

To date, iTunes is the best exemplar of this principle. This application seamlessly reaches from the handheld device to a massive web back-end, with the PC acting as a local cache and control station. There have been many previous attempts to bring web content to portable devices, but the iPod/iTunes combination is one of the first such applications designed from the ground up to span multiple devices. TiVo is another good example.

iTunes and TiVo also demonstrate many of the other core principles of Web 2.0. They are not web applications per se, but they leverage the power of the web platform, making it a seamless, almost invisible part of their infrastructure. Data management is most clearly the heart of their offering. They are services, not packaged applications (although in the case of iTunes, it can be used as a packaged application, managing only the user's local data). What's more, both TiVo and iTunes show some budding use of collective intelligence, although in both cases, their experiments are at war with those of the intellectual property lobby. There's only a limited architecture of participation in iTunes, though the recent addition of podcasting changes that equation substantially.

This is one of the areas of Web 2.0 where we expect to see some of the greatest change, as more and more devices are connected to the new platform. What applications become possible when our phones and our cars are not consuming data but reporting it? Real-time traffic monitoring, flash mobs, and citizen journalism are only a few of the early warning signs of the capabilities of the new platform.

7. Rich User Experiences

As early as Pei Wei's Viola browser in 1992,[17] the web was being used to deliver "applets" and other kinds of active content within the web browser. Java's introduction in 1995 was framed around the delivery of such applets. JavaScript and then DHTML were introduced as lightweight ways to provide client-side programmability and richer user experiences. Several years ago, Macromedia coined the term "Rich Internet Applications" (which has also been picked up by open-source Flash competitor Laszlo Systems) to highlight the capabilities of Flash to deliver not just multimedia content but also GUI-style application experiences.

However, the potential of the web to deliver full-scale applications didn't hit the mainstream until Google introduced Gmail, quickly followed by Google Maps, web-based applications with rich user interfaces and PC-equivalent interactivity. The collection of technologies used by Google was christened "AJAX," in a seminal essay by Jesse James Garrett of web-design firm Adaptive Path. He wrote,

> Ajax isn't a technology. It's really several technologies, each flourishing in its own right, coming together in powerful new ways. Ajax incorporates:
>
> - standards-based presentation using XHTML and CSS;
> - dynamic display and interaction using the Document Object Model;
> - data interchange and manipulation using XML and XSLT;
> - asynchronous data retrieval using XMLHttpRequest;
> - and JavaScript binding everything together.[18]

AJAX is also a key component of Web 2.0 applications such as Flickr, now part of Yahoo!, 37signals' applications basecamp and backpack, as well as other Google applications such as Gmail and Orkut. We're entering an unprecedented period of user-interface innovation, as web developers are finally able to build web applications as rich as local PC-based applications.

Interestingly, many of the capabilities now being explored have been around for many years. In the late '90s, both Microsoft and Netscape had a vision of the kind of capabilities that are now finally being realized, but their battle over the standards to be used made cross-browser applications difficult. It was only when Microsoft definitively won the browser wars, and there was a single de facto browser standard to write to, that this kind of application became possible. And while Firefox has reintroduced competition to the browser market, at least so far we haven't seen the destructive competition over web standards that held back progress in the '90s.

We expect to see many new web applications over the next few years, both truly novel applications and rich web reimplementations of PC applications. Every platform change to date has also created opportunities for a leadership change in the dominant applications of the previous platform.

Gmail has already provided some interesting innovations in e-mail,[19] combining the strengths of the web (accessible from anywhere, deep database competencies, searchability) with user interfaces that approach PC interfaces in usability. Meanwhile, other mail clients on the PC platform are nibbling away at the problem from the other end, adding instant-messaging (IM) and presence capabilities. How far are we from an integrated communications client combining the best of e-mail, IM, and the cell phone, using Voice over Internet Protocol (VoIP) to add voice capabilities to the rich capabilities of web applications? The race is on.

It's easy to see how Web 2.0 will also remake the address book. A Web 2.0–style address book would treat the local address book on the PC or phone merely as a cache of the contacts you've explicitly asked the system to remember. Meanwhile, a web-based synchronization agent, Gmail style, would remember every message sent or received and every e-mail address and every phone number used and would build social networking heuristics to decide which ones to offer up as alternatives when an answer wasn't found in the local cache. Lacking an answer there, the system would query the broader social network.

A Web 2.0 word processor would support wiki-style collaborative editing, not just standalone documents. But it would also support the rich formatting we've come to expect in PC-based word processors. Writely is a good example of such an application, although it hasn't yet gained wide traction.[20]

Nor will the Web 2.0 revolution be limited to PC applications. Salesforce.com demonstrates how the web can be used to deliver software as a service, in enterprise-scale applications such as Customer Relations Management (CRM).

The competitive opportunity for new entrants is to fully embrace the potential of Web 2.0. Companies that succeed will create applications that learn from their users, using an architecture of participation to build a commanding advantage not just in the software interface but in the richness of the shared data.

Core Competencies of Web 2.0 Companies

In exploring the seven principles discussed in this essay, we've highlighted some of the principal features of Web 2.0. Each of the examples we've explored demonstrates one or more of those key principles but may miss others. Let's close, therefore, by summarizing what we believe to be the core competencies of Web 2.0 companies:

- Services, not packaged software, with cost-effective scalability
- Control over unique, hard-to-re-create data sources that get richer as more people use them
- Trusting users as codevelopers
- Harnessing collective intelligence
- Leveraging the long tail through customer self-service
- Software above the level of a single device
- Lightweight user interfaces, development models, *and* business models

The next time a company claims that it's "Web 2.0," test its features against this list. The more points it scores, the more it is worthy of the name. Remember, though, that excellence in one area may be more telling than some small steps in all seven.

NOTES

This chapter was originally published on September 30, 2005, on *O'Reilly Radar* (blog), at http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.htm (accessed July 17, 2010). This work is licensed under the Creative Commons Attribution-ShareAlike license.

1. Carlota Perez, *Technological Revolutions and Financial Capital: The Dynamics of Bubbles and Golden Ages* (Cheltenham, UK: Elgar, 2002).
2. The first Web 2.0 Conference took place October 5–7, 2004, in San Francisco.
3. Tim O'Reilly, "Not 2.0?," *O'Reilly Radar* (blog), August 5, 2005, http://radar.oreilly.com/archives/2005/08/not-20.html (accessed July 17, 2010).
4. Eric S. Raymond, *The Cathedral and the Bazaar* (Sebastopol, CA: O'Reilly, 2001), http://catb.org/esr/writings/homesteading/cathedral-bazaar/ar01s05.html (accessed July 17, 2010).

5. Thomas Vander Wal, "Folksonomy," *vanderwal.net* (blog), February 2, 2007, http://vanderwal.net/folksonomy.html (accessed July 17, 2010).

6. Yochai Benkler, "Coase's Penguin, or, Linux and the Nature of the Firm," *Yale Law Journal* 369 (2002), available online at http://www.benkler.org/CoasesPenguin.html (accessed July 17, 2010).

7. Rich Skrenta, "The Incremental Web," *topix* (blog), February 12, 2005, http://blog.topix.com/2005/02/the-incremental-web.html (accessed July 17, 2010).

8. Mark Pilgrim, "Dive into XML: What Is RSS," XML.com, December 18, 2002, http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html (accessed July 17, 2010); World Wide Web Consortium, "RSS 2.0 Specification," Fall 2002, http://validator.w3.org/feed/docs/rss2.html (accessed July 17, 2010).

9. Tom Coates, "On Permalinks and Paradigms . . . ," *Plasticbag.org* (blog), June 11, 2003, http://www.plasticbag.org/archives/2003/06/on_permalinks_and_paradigms/.

10. James Surowiecki, *The Wisdom of Crowds* (New York: Doubleday, 2004).

11. Dan Gillmor, *We the Media* (Sebastopol, CA: O'Reilly, 2004), http://www.authorama.com/we-the-media-8.html (accessed July 17, 2010).

12. Tim O'Reilly, "Hardware, Software, and Infoware," in *Open Sources: Voices from the Open Source Revolution*, ed. Chris DiBona, Sam Ockman, and Mark Stone (Sebastopol, CA: O'Reilly, 1999).

13. Tom Coates, "Cal Henderson on 'How We Built Flickr . . . ,'" *Plasticbag.org* (blog), June 20, 2005, http://www.plasticbag.org/archives/2005/06/cal_henderson_on_how_we_built_flickr/ (accessed July 17, 2010); Chris May, "Deploy Every 30 Minutes: Redux," *Secret Plans and Clever Tricks* (blog), August 25, 2005, http://blogs.warwick.ac.uk/chrismay/entry/release_once_every/ (accessed July 17, 2010).

14. Phil Wainewright, "Why Microsoft Can't Best Google," *Software as Services* (blog), ZDNet, August 24, 2005, http://www.zdnet.com/blog/saas/why-microsoft-cant-best-google/13 (accessed July 17, 2010).

15. J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End Arguments in System Design," *ACM Transactions on Computer Systems* 2, no. 4 (1984): 277–288, http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf (accessed July 17, 2010); M. S Blumenthal and D. D. Clark, "Rethinking the Design of the Internet: The End to End Arguments vs. the Brave New World," *ACM Transactions on Internet Technology* 1, no. 1 (2001): 70–109.

16. David Stutz, "Advice to Microsoft Regarding Commodity Software," *The Synthesist* (blog), February 11, 2003, http://www.synthesist.net/writing/onleavingms.html (accessed July 17, 2010).

17. Viola Web Browser, http://www.viola.org/ (accessed July 17, 2010).

18. Jesse James Garrett, "Ajax: A New Approach to Web Applications," Adaptive Path website, February 18, 2005, http://www.adaptivepath.com/ideas/essays/archives/000385.php (accessed July 17, 2010).

19. Tim O'Reilly, "The Fuss about Gmail and Privacy: Nine Reasons Why It's Bogus," *O'Reilly Net* (blog), April 16, 2004, http://www.oreillynet.com/pub/wlg/4707 (accessed July 17, 2010).

20. *Editor's note*: Writely.com was purchased by Google on March 6, 2006, and has subsequently become Google Docs.