By Aaron Weiss

WebOS: Say Goodbye to Desktop Applications

Some call it the second coming of the dot-coms. The blogosphere is buzzing, and new buzzwords are minted daily in the press: "Web 2.0," "Weblications," and "WebOS"—the Web-based Operating System. Email, the Internet's first killer app, has now largely migrated from the desktop to the Web. Instant messaging, word processing, and even spreadsheets aren't far behind. Something is happening here, as Bob Dylan once sang, but do we know what it is?

Illustration by Steven Adler

The Web is making an evolutionary leap from a document-delivery system to an application framework. The new Web has the potential to pull the rug out from under the desktop, along with the applications and vendors who have built their fortunes there. Of course, not everyone is convinced that evolution equals revolution. Critics say the WebOS is more hot air than hot news. But WebOS evangelists believe they know exactly what's happening here—and that it has already begun.

The WebOS Vision

The entirety of WebOS isn't a real thing (yet), nor is it a specific thing (yet). WebOS is a concept which lays down a roadmap—a gauntlet, even—for how a convergence of software and the Web may up-end the way we compute and disrupt entrenched market forces. The history of personal computing has so far been dominated by one-box solutions—a physical container holding all the software and hardware you need to run your applications. A desktop operating system like Microsoft Windows or Apple's OS X provides a series of interfaces between the hardware inside the box and the software you run—your word processors, graphic design tools, Web browser, etc. The desktop operating system expects most of its resources to be located inside that box—the graphics card to output video, the hard disks to store data, and so on.

The Internet, and particularly the Web, exploded the concept of computers as isolated islands. We are now connected and interconnected. Why, then, limit the operating system to one box? The WebOS vision explodes the operating system itself, outside the physical bounds of a box, and across the Internet. Why limit file storage to local disks? Why limit application logic to local processing on your CPU? Why limit access to documents to just one person at a time? In other words, why not erase the boundaries between your local computer and the whole global network?

Déjà Vu All Over Again

To some veterans of the technology industry, the excitement bubbling over WebOS sounds more than a little familiar. Flashback to the '70s, when central processing units were housed in their own rooms, and computer scientists and grad students huddled around "dumb terminals" to write code in Fortran. Dumb terminals were simply monitors and keyboards, relying on processing power elsewhere (the next room, usually), since it was not feasible to equip each workstation with its own processing unit. Even by the early '80s, as the personal computer revolution was just beginning to catch fire, dumb terminals ruled, although by then young computer scientists had the luxury of their Sony Walkman to keep them company.

Historians—that is, those of us older than 20—will also remember the short-lived "thin client" push during the dot-com boom of the mid-'90s. Espoused most publicly by Oracle chief Larry Ellison, thin clients represented a vision of the WebOS to come. Selling cheap hardware boxes that relied on Internet connectivity for most of their productivity was hailed as the future of personal computing. But the vision was much too early. Thin client advocates made the mistake of exchanging power for connectivity. Thin client machines themselves were underpowered and kind of fey, particularly in a market where full-muscled PC hardware was rapidly declining in price. More importantly, thin clients weren't mobile, just when mobile computing was taking off. And the Internet itself did not reach a critical mass of ubiquity nearly as quickly as thin client investors had assumed. In other words, almost everything that could go wrong with the thin client initiative did go wrong. Believers in the WebOS see the failed thin client movement not as philosophically misguided, necessarily, but ill-timed and poorly executed.

So it's not so surprising that in 2005, college students queued up at workstations to log into their Web-based email while tethered to their iPods might not look like such a revolution, especially to those who have been around the block. In fact, it could almost appear that we've come full circle—after empowering ourselves for 20 years with personal computers that packed several times the power of a '70s mainframe into a small box, we're finally back to dumb terminals?

Not exactly. Remote computing in the dumb terminal era was born of necessity. Today's WebOS revolution is being driven, perhaps more than anything else, by mobility—particularly among today's professional workers and youth culture. By 2005, laptop computer sales outstripped desktop PC sales for the first time in history, with over 53 percent of the total PC market. In the US, broadband penetration exceeds 50 percent at home and 70 percent at the workplace—numbers which are even higher in urban areas, and significantly higher—over 80 percent—in smaller technologically-advanced countries such as South Korea and Japan.

With urban and suburban populations increasingly saturated with computing and connectivity, the cord has been cut between an individual and a particular machine. Computing surrounds us, and our technological lives are rapidly becoming abstracted from hardware, with computers themselves simply becoming access points to our online lives. In light of this, WebOS seems inevitable.

Be Here Now

Inevitable? It's already here. It is currently fashionable to proclaim Google's Gmail the pioneer of WebOS: the service that brought weblications to the masses, the iPod of Web 2.0, if you will. Putting aside the buzzword overkill for a moment, application-like behavior on the Web has been brewing and evolving for at least ten years. One of the most fascinating aspects of the WebOS story is how it's come along without anybody really planning for it.

The Web, and its underlying protocols HTTP and HTML, were designed as a document delivery system. Hyperlinks were the big twist that had everyone excited about the Web in the early '90s. HTTP transports documents across the network and HTML describes how to render them on the screen. By today's standards, the system seems almost quaint.

It didn't take long for enterprising engineers, professional and hobbyist alike, to figure out that the Web could do more than deliver static documents. A technology called CGI—the Common Gateway Interface—was cobbled together for Web servers, which let them run programs to create results. Now, rather than simply ask a server to "give me the document called XYZ," Web users could ask for data to be generated on demand. "Give me a climate report for every Tuesday last year"—and the server could launch a process to figure out the answer, and deliver that answer to the browser. They called them dynamic Web pages—pages put together using a wide range of resources, from queries

IT DIDN'T TAKE LONG FOR ENTERPRISING ENGINEERS, PROFESSIONAL AND HOBBYIST ALIKE, TO FIGURE OUT THAT THE WEB COULD DO MORE THAN DELIVER STATIC DOCUMENTS.

WHAT HAS COLLAPSED IS A LIMITATION—ONE THAT KEPT CLIENT AND SERVER COMMUNICATING ONLY IN FORMAL, VERBOSE, ROUND-TRIP EXCHANGES.

to databases to internal calculations and anything else to which the server has access.

CGI programs became increasingly sophisticated, and eventually spawned a slew of server-side programming languages designed to generate Web page results triggered by user requests. We're barely past 1996 at this point and the Web has already moved beyond simple document delivery.

Browsers were very limited front-ends, however, since they didn't offer the user a whole lot of controls other than hyperlinks and some form fields. Also, browsers were entirely "dumb"—they did no processing of their own. This changed with the introduction of JavaScript, which added client-side programming to the browser. Now, Web pages could contain code to tell the browser to do stuff—calculate values, pop up windows, make sounds. Web pages could deliver miniature applications to the browser, rather than simply text documents.

Technologies like CGI and JavaScript were really introduced in the field—meaning, rather than be coming from a top-down, centralized planning body, they first appeared in practice. People, and businesses, invented them because they needed them. The official standards body for the Web—the World Wide Web Consortium, or W3C—often found itself playing catch-up. The Web was being extended through sheer force of will, rather than by design.

So, are weblications new? Even five years ago you could visit MapQuest and generate custom driving directions. Was that a weblication? It was, which begs the question—what's so new right now? Why the buzzwords? Why this article? The answer is that a great wall has collapsed.

Imagine a Web page asks a Web server to calculate, say, a metric-to-imperial conversion. Assuming the Web server was programmed to handle this request, no problem. Except that the server can't return just the answer—say, 42.7. What would the browser do with that? The browser only knows how to receive Web pages. So, the server has to deliver a whole Web page containing the results.

The process is like negotiating when to meet your friend tonight for dinner, written out entirely in longhand. "Dear Sidney," you begin, and make your request, complete with dated greeting and proper signature. Sidney replies with a formal letter. You reply with another formal letter. And so on. There are inevitably delays in transporting your letters back and forth. The whole process is slow and cumbersome. Now transplant this exchange to text messaging on a cell phone. "7 ok?" "can't, 8?" "ok! Cya!" Done.

What has collapsed is a limitation—one that kept client and server communicating only in formal, verbose, round-trip exchanges. Now, client and server can exchange chatter, moving tidbits of data back and forth, reacting to them in real-time. This highly technical but extremely influential shift is called asynchronous communication. To use a popular example, say you are registering for a site and you are asked to choose a login name. Typically you would fill out the form, click some kind of "submit" button, and await a response. Should your chosen name be unavailable, the server would deliver a new page asking you to try again.

Using asynchronous communication, the registration form can instantly validate your login name against the server. No need to

round trip the whole page. If the server says no-go, the browser can tell you so immediately. It might even suggest available alternatives. In other words, a Web page can now behave just like desktop application, rather than like two far-flung correspondents exchanging letters. But where the desktop executes both the front and back end of a program, a Web application executes the front end on the client, and the back end on the server, wherever that may be.

Where did Web-based asynchronous communication come from? Microsoft, actually. They introduced the technology in Internet Explorer 5, with a programming feature called XMLHTTP built on their ActiveX technology. Developers at the open-source Mozilla organization created their own emulation of Microsoft's new object, known as the XMLHttpRequest Object. To fully leverage the technology and build desktop-like Web-based applications, XMLHttpRequest is used in conjunction with other pre-existing Web development technologies. Together, the constellation has come to be known informally as AJAX—Asynchronous JavaScript and XML.

Like CGI and JavaScript before it, AJAX has not come down from an official standards body. Indeed, some Web developers reject AJAX on the basis that neither it nor the XMLHttpRequest object is standard—they have not been approved by the W3C. Yet, AJAX development is proliferating in the field, powering many of the Web-based applications that are behind the WebOS buzz.

AJAX or Not

GMail is an AJAX Web application. So are Google Maps and Google Suggest. Google's endorsement of AJAX development is good enough for many developers, and may solidify it as a de facto standard, until official standards bodies catch up yet again. But whether or not AJAX technology is standardized matters little—the breakthrough for WebOS is not the specific cluster of technologies in vogue now, but that they are being used at all. Clearly, asynchronous communication on the Web is here to stay, regardless of which technologies are used. And that means Web-based applications are here now, maturing and evolving every day.

When Gmail was introduced, it took a swipe at Yahoo and Hotmail, the incumbent Web-based email providers. Gmail offered both an increase in storage space and an advanced interface. Yahoo and Hotmail are now about to strike back, with impending launches of their own AJAX-based email. Yahoo is generating major buzz with its soon-to-be-released WebOS email interface, which looks and behaves very much like Microsoft Outlook on the desktop. Users can drag-and-drop messages into folders, enjoy address-book-based auto-completion when composing new messages, view multiple messages in a tabbed layout, and sort and search seemingly on the fly.

While competition among the big boys legitimizes and popularizes Web-based applications, many smaller developers are working in the trenches to expand their scope well beyond email.

Don't have an instant-messaging client handy? Visit meebo.com, an AJAX-based IM interface for the Web. With just a browser you can sign on to your AOL, Yahoo, GTalk, or MSN IM, and chat with your buddies through an interface virtually identical to desktop IM.

The hottest area for Web-based application development is office productivity. Want to write, save, and access documents from anywhere? gOffice features an online word processor. As does Writely. Or SynchroEdit, with which several people can collaborate simultaneously online. Or the FCKEditor, an open source Web-based word processor developers can integrate into any site. All feature Word-like composition screens, with on-the-fly formatting tools like styles, fonts,

alignment, and many of the goodies we've been using to format documents on the desktop for two decades.

Then there's Kiko, a Web-based shareable calendar that can read iCal/Outlook formats, generate RSS data, and is of course accessible globally.

Or, for a non-AJAX twist on a Web-based application, ThinkFree Office relies on Java to deliver a Web-based Office suite, with word processing, presentation, and spreadsheet applications.

Smaller applications, too numerous to list, are filling out the WebOS at the edges—bookmark managers, HTML editors, and map generators. Experimental projects like Openomy, an online filesystem, are laying the groundwork for less visible WebOS foundations.

Even the red hot online music market is moving to the Web. Apple's iTunes proved that selling legitimate digital music online is a viable, profitable business model. But iTunes works by downloading music to the buyer, who listens to it offline—from their desktop, iPod, or other audio device. In contrast, competitors Napster, RealNetworks, and America Online have all announced plans to offer Web-based music services. Customers will stream their music directly from the vendors' sites, in effect creating an iTunes-on-the-Web.

Regardless of which technologies they use, developers are rushing headlong into Web-based application development. Some are undoubtedly motivated by the WebOS' wide-open possibilities. Others, like gold-rushers before them, are hoping to establish a foothold in a burgeoning market.

An Eruption of Disruption

On the desktop, applications run on top of the operating system. But Web-based applications run on top of the browser. Doesn't that make the browser an operating system? In a way, it does—which is exactly the WebOS vision. Of course, today's Web browsers are still bound to some extent to their underlying operating system. Internet Explorer is designed to function in a Windows environment. But as cross-platform browsers like Firefox proliferate, reliance on the operating system becomes less important. The operating system is increasingly relegated to handling low-level functions, like writing data to the hard disk.

The problem for operating system vendors like Microsoft and Apple is that nobody buys an operating system because of how it writes data to the hard disk. Throughout the era of the personal computer, the operating system has defined the identity of a system. Even though we ultimately use a computer for its applications, Apple has enjoyed great success creating customer loyalty through the design and personality of its operating system. Microsoft, of course, has achieved a dominant position in the industry through widespread adoption of Windows.

So what if you didn't need Windows? What if you managed your email through your browser, wrote your reports and documents through a browser-based word processor, saved your data on remote servers, played games through Web sites—what would you need your operating system for?

Most everyone agrees this vision of WebOS is years away. For some, it is a vision of utopia—the ultimate technological liberation. To others, WebOS is a vision of opportunity, the chance to generate new revenues in markets where old incumbents have reigned. And to those incumbents, the WebOS vision could be something of a nightmare.

For Microsoft in particular, WebOS presents a potential threat to the company's very core. In 2000 Microsoft unveiled their much-heralded "dot Net" strategy, devised in part to anticipate the possible rise of a non-Microsoft-based WebOS. Dot Net was a broad tent encompassing a variety of technologies, ultimately aiming to capture the network-based operating system before it could

MAJOR SOFTWARE VENDORS HAVE LONG DABBLED WITH THE IDEA OF SHIFTING THE SOFTWARE BUSINESS TO A RENTAL OR SUBSCRIPTION MODEL, WHICH WEBOS FACILITATES.

mature elsewhere. While dot Net still exists today, it has been absorbed into the array of tools available to Windows and Web developers. It did not define Web development. The Internet proved larger than Microsoft's strategy. And although today's Web-based applications based on AJAX techniques evolved from Microsoft's XMLHTTP innovation, the tools now exist in the wild, open and free from any one company's control.

WebOS presents a challenge to a market leader like Microsoft. How do they aggressively position themselves in an emerging market without cannibalizing their current business models? Microsoft Office, for example, currently enjoys an over 90-percent market share in the office-suite market. Giving away free access to Web-based productivity tools could undermine that business. But allowing others to enter the Web application space and establish their own tools could be even worse.

To address the challenge, Microsoft announced in November plans to release "Windows Live" and "Office Live"—Web-based platforms aimed at consumers and small businesses. Both are said to provide some of the functions for which users typically turn to their desktop: email, instant messaging, and word processing. Plans for a tiered pricing structure include advertising-supported free access, a sign that Microsoft is being pushed into new and unfamiliar territory—a business model led by Google.

To current and future Microsoft competitors, the WebOS is all upside. Google is keenly aware of this, and is proving it with their avalanche of Web-based product releases. Microsoft is clearly keen to battle back, but risks upsetting their established apple cart. Which is precisely why WebOS is being described as a potentially disruptive technology. Perhaps disruptive is too polite a word. WebOS could be an earthquake.

Is this the big one? Not everyone thinks so.

Whoa There

Critics of both Web-based applications and AJAX techniques in particular suggest that enthusiasts might be getting a little ahead of themselves.

Privacy advocates, for example, wonder how comfortable people would be turning over their work for storage on remote servers. Would you trust third parties to securely store your documents, spreadsheets, and other files? Yet, WebOS advocates argue that competition among Web-based application vendors will lead them to provide better backup and security than you have at home (and maybe the office). And besides, they say, millions have flocked to Web-based email, despite these very same concerns.

Another concern raised is whether Web-based applications are really a Trojan horse for transforming software licensing from product to service. For years, major software vendors have openly dabbled with the idea of shifting the software business to a rental or subscription model, guaranteeing ongoing revenues and eliminating the need to force and cajole users into upgrades they may not be motivated to buy. Whether or not today's WebOS developers buy into this, their efforts could grease the path toward software sub-

scriptions. The counterargument suggests that WebOS will increase competition, letting the marketplace decide. All WebOS applications would be on equal footing, because the traditional obstacles of distribution are eliminated—meaning, in theory, more players, and more licensing models to choose from.

WebOS applications, though, could be said to remove control from the end user. Some see this as offering positives and negatives. On the positive side, because WebOS applications are essentially vertical, they can be kept always up to date, for all users everywhere at once. If the vendor fixes a bug in a WebOS application, everyone benefits instantly. Everyone uses the same version at the same time. This could lead to reduced costs in product support and end-user fees. But critics warn, this also puts complete control in the vendor's hands at all times. If you don't renew your subscription, do you lose access to your data? What if you did renew, but the payment was incorrectly processed? What if the vendor decides to eliminate a feature you relied upon for the product's value? A desktop application isn't going to slide out from under you. You are free to buy and use a copy of Microsoft Word 97 for all eternity, and it will never change.

Those with a fresh memory for the failure of thin clients might ask, isn't it still too soon? Broadband penetration, especially in the US, is far from universal, and the broadband we do have isn't that great. It's certainly no substitute for the responsiveness of a three-gigahertz PC on your lap.

Numerous critics also say that AJAX and related techniques in vogue today for building Web applications aren't good enough. They aren't standardized, they rely on browser gimmickry, stretching browsers—and the Web in general—way beyond what they were designed for. Put another way, today's Web applications are pushing our current technologies beyond their tolerances, with weak tools nowhere near as robust or mature as those available for developing desktop applications. HTML, they say, is a cruddy interface language, and AJAX just a duct-tape approach to cramming a motley collection of tools together.

Ends and Means

But WebOS advocates might say, "Who cares?" WebOS isn't about today's tools—AJAX or Java or whatever. Those will mature. They will become standardized. It isn't about today's slow and spotty broadband. What matters about WebOS is not the tech used to get there, but why people want to use WebOS software.

It's already been proved that users love the results. Web-based email, after all, could be described as technologically inferior to desktop email on many levels. End users don't seem to care. Because the value we're looking for today, in a mobile, connected world, is in a service's ubiquity—not its underlying framework, or whether the code is elegant, or based on standards. Simply that the service is here, there, and everywhere.

End users, it seems, are willing to put up with all the other deficiencies. At least, for email. But is email just a special case, or a divining rod pointing us toward the promise of WebOS?

You don't have to throw a rock very far to hit an extreme opinion somewhere on the Internet—that the WebOS is nothing but hype, it's the next dot-com bubble, or that it's the savior of mankind and the future of all computing. WebOS may be none of those things. How disruptive will it become? Google is banking heavily on it, and Microsoft's recent actions indicate they're onto Google's plan. What will they do about it? Those answers aren't yet known, but the questions prove that WebOS is a real market and we are, in fact, already immersed in it. ~

Aaron Weiss *is a technology writer and Web developer in upstate New York: www.bordella.com.*