Select Font Size: A A A

The Web within the Web

By: **Enrique Castro-Leon**

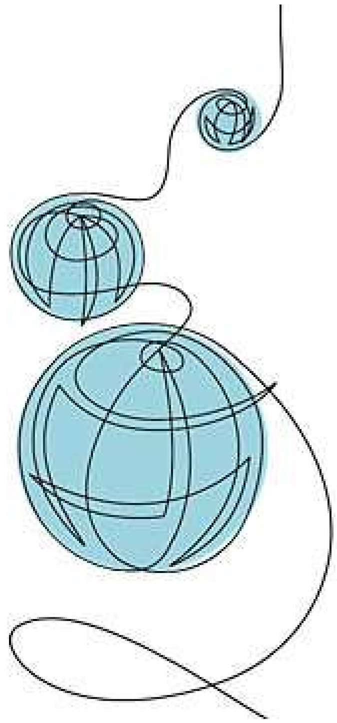


ILLUSTRATION: FELIX SOCKWELL

By Many Measures, the Web is a phenomenal business success. Entire industries, such as auctions, book selling, travel reservations, news dissemination, and classified advertising, have undergone sea changes because of it. Web companies like eBay, Amazon.com, Yahoo!, and Monster are huge enterprises. Some of them are even turning a profit.

And yet the Web has also been an abject failure, at least so far. According to one estimate, e-commerce revenue for 2002 was a paltry 3.2 percent of all U.S. commercial transactions (about US $72 billion out of $2.25 trillion). Briefly high-flying companies like Priceline.com continue to disappoint investors. And Time Warner has dropped its Internet half, AOL, from its name, perhaps as a precursor to severing the company itself.

Even within the industries that the Web is coming to dominate, such as travel bookings and music sales, there's so much more that could be done. The reason is, in a word, databases: dusty, musty databases filled with useful data that would be far more useful if linked with other, equally dusty databases; enormous databases that are locked up inside ancient mainframes and quaintly archaic minicomputers; lonely databases residing on specialized file servers throughout an enterprise; even modern databases on Web servers, all dressed up and ready to go, but stuck in long-obsolete proprietary formats or accessible only through hypermodern scripting languages.

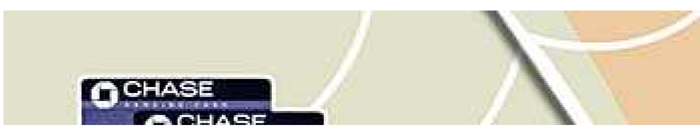
Second-generation e-commerce will depend on unlocking those databases. And it's starting to happen, thanks to a combination of modest technologies that together go by the name of Web services. Web services are a way programmers can make their databases available across the Web, let other programmers access them, and tie these disparate databases together into services that are novel, perhaps even wonderful. Travel agents could put together vacation packages from various airlines and hotel chains, extracting the best seasonal discounts automatically. A Web servicesbased wedding registry wouldn't be limited to just one department store.

Beyond these consumer applications, over the next few years Web services will spur a transformation—albeit a quiet one—within a number of much bigger industries, including some, like insurance and banking, that haven't really had a kick-start since the 1964 introduction of the IBM 360 mainframe.

Web services already are brushing the cobwebs off some of those old databases. Take, for example, air travel, an industry whose databases have always been a bit more open than most, because of the need to interact with independent travel agents. Until recently, the Internet had changed air travel only in tiny ways. For example, you could book your ticket online and then walk up to the check-in counter with nothing more than a printout of an e-mail message. But little else about the check-in process—including its long queues—had changed much in 30 years. Now, though, new self-service check-in kiosks are turning up, based, as it happens, on Web services software "


DATABASES
PCs
Reservation systems


INTERFACES
Cellphones


CHASE

WEB SERVICES FLY HIGH: Air travel is one industry being dramatically changed by Web services. When you fly somewhere, you can make or change a reservation or check in for a flight simply by identifying yourself to the services system via one of several interfaces [right You can swipe a bank card or credit card at a kiosk or key identification information into a PC, PDA, or cellphone. Behind the scenes [left], a query goes to a credit card clearinghouse, which sends your identifying information to the Web services application. If that clearinghouse is inaccessible, the query goes to alternative clearinghouses until it gets a response. Making or changing a reservation draws information from a number of databases, including airline reservation systems and, if you have one, a frequent flyer account. The system can also check a database of arrivals and departures and tell you whether your flight is on time.

FELIX SOCKWELL, MICHAEL R. VELLA

"

The kiosks let you identify yourself by, say, swiping a credit card or entering your e-ticket information. You can confirm your flight itinerary, and then some surprising screens appear. For example, there's a seating chart, with your previously assigned seat highlighted in yellow and still-available seats in green. If you see one you'd prefer, just touch it with your finger—voilà, that's your new seat assignment. At the end, you print out a boarding pass and head for the gate. Unfortunately, the new software can't do anything about that line.

What has changed? The airline has opened up its databases to organizations and the public, showing you data hitherto seen only by airline personnel and travel agents. There's something almost sinfully pleasurable about reviewing an airline's seating chart, as if you were able to look over the shoulder of the airline agent at the counter. But what has changed behind the screen? The differences are a new layer of software—Web services—which link the airlines' seating database to the easy-to-navigate kiosk interface.

Web services aren't just about pouring old database wine into new bottles. They take databases—new ones as well as old—and place them in modern, easy-to-use decanters, whether it's a Web browser, a custom application, such as the airline check-in kiosk, or just plain e-mail. Web services open these closed database applications and let them breathe. Once opened, a database can be used by other departments within an enterprise, vendors and customers, and, as in the airport kiosk, the public at large.

Travel is changing in other ways than the airport check-in. Galileo, the giant on- and off-line travel service from Galileo International Inc., in Parsippany, N.J., is using Web services to streamline back-office operations. For instance, it's consolidating the large number of individual processes involved in a single transaction like buying a ticket or claiming a frequent flyer award. One of Galileo's main goals is appealing: to do a better job of combining flights with empty seats, hotels with empty rooms, and rental cars sitting in garages into last-minute vacation specials. If it works, it could make a lot of money for the airlines, hotels, and car rental agencies—and for Galileo, of course.

Everything from online shopping and auctions to financial services stands to benefit. Take, for example, the insurance industry, a strong contender in the oldest-and-mustiest-database competition. For years and years, Joe Salesguy has paid Jane Customer a call, scrawling his notes on little slips of paper. Eventually, Joe goes back to the office, spills out all the paper onto a desk, and updates his prospective-client lists. Instead, a Web service could let him update the database with a text message from his cellphone. Not only would he save the trip to the office, but his manager would also be able to see Joe's day progressing in real time. When Joe finally makes the sale, he would be able to create a new account record just minutes later, using an ordinary Web browser on his laptop or at an Internet cafe.

Having finally bought a policy, Jane might want to arrange a direct monthly payment from her bank to the insurance agency. Web services could come into play yet again, as long as either of those financial institutions, the bank or the insurance company, opened up its database to the other (with proper security built in, of course). If several insurance companies opened up their catalogs of insurance policies to Web services, an independent insurance agent could write a software application that compared them and helped customers like Jane find the best policy for their needs.

The insurance companies wouldn't even need to be aware that such a third-party application had been created, because the Web services application would pluck data from publicly available databases found through the insurance companies' Web sites. No wonder Dmitri Tcherevik, vice president of Web services management for Islandia, N.Y.based Computer Associates International Inc., predicts that financial institutions will be "the biggest beneficiaries of Web services."

Many customers will prefer the new system, whether they're checking in at the airport more quickly or updating a sales contact database while waiting for a double latte at a Wi-Fienabled coffee shop. But in addition to making users happy, Web services also dramatically drive down costs. Ron Schmelzer, a senior analyst for ZapThink LLC in Waltham, Mass., a Web services consulting firm, says that he's seen software project costs cut by 90 percent because of Web services.

The largest savings come from reusing software from project to project. Think of an application, such as selling an insurance policy, as consisting of several layers. The user interface, at the top, will be different for a PC or a cellphone. The bottom layer, in which data is extracted, will differ for each database being drawn from. But the entire middle layer, in which the data is processed and prepared for presentation to the user, can be essentially the same. "The greatest integration costs are in middleware," says Schmelzer.

The consequences—reducing costs, adding revenue from last-minute sales, and having better back-office operations, more-efficient salespeople, and happier customers—may be even greater than the changes wrought a decade ago when businesses first started using the Web to augment their sales, inventory, and other systems.

Take e-commerce, which has matured, but only up to a point. Sure, we can now get real-time bank account balances, transfer money between accounts, and pay bills; we can shop with online catalogs and track our purchases from the privacy of our homes. But almost everything we do on the Web today, including Google searches, catalog shopping, and looking up driving directions, can be done only with a human sitting in front of a screen.

Nevertheless, we've certainly progressed from the pre-Internet era. What we've built is, in fact, the foundation for the new world of Web services. In 10 short years, Web browsers have liberated us from the tyranny of specific hardware and the near-monopoly of the Windows operating system. Internet Explorer, Netscape, Opera, Safari—they all display Web pages in more or less the same way, regardless of the platform they're running on: Microsoft Windows, Mac OS 9 or X, Linux or Unix. That's because of two things: the Hypertext Transfer Protocol, which provides a standard for the way Web pages are downloaded from a Web site to a computer, and the generic nature of Web pages themselves.

Once a page is coded in Hypertext Markup Language (HTML), a browser knows just how to display it, with markup codes specifying fonts, heading styles, columns, table structures, location on the screen of a graphical image, and so on. But HTML was designed to encode things that will be viewed by people, rather than processed by another machine. HTML mixes formatting commands (such as color and positioning) with data (the text itself, graphics, sounds, and so on), because it was designed as a display language.

Ferreting through HTML to retrieve embedded nuggets of data while simultaneously ignoring formatting constructs isn't impossible, but it's unnecessarily difficult. This parsing task is complicated by the fact that HTML code is not static. It changes, for instance, whenever a Web site owner changes the appearance of a Web page. Some changes are built into a Web site's very design: a bank customer, for example, will see a different screen depending on whether or not there are enough funds to cover the withdrawal. Furthermore, HTML coding often contains errors; those mistakes can easily trip up a parsing program.

So if Web services are to build powerful networks of collaborating databases and services, the first step is replacing HTML with something more compatible with the world of databases, something that can be understood by another computer. And such a new language has been developed. It's a superset of HTML, called XML, for Extensible Markup Language.

XML is a universal standard for representing data, so XML-based programs are inherently interoperable. Basically, XML uses the lowest common data denominator available, which is text. Here's how it works: data in XML form is consigned to specific fields. There might be one field for "price," for example, and another for "quantity."

Older mainframes get a new lease on life

Once information is in XML form, it can be extracted from different databases and compared, so long as the two databases have equivalent fields, such as price and quantity. But what if the databases have fields that are similar but not equivalent? It would be a problem today, but perhaps not tomorrow. Emerging Web service innovations would add extra data, called metadata, that would let a database "announce" its structure. Then two different databases with similar fields could be compared by a software program with no human intervention at all.

Since Web services are used to create interoperable Web applications, there must be some mechanism to move XML data across the Internet. The easiest way would be to take advantage of an already existing protocol, the obvious candidate being the Hypertext Transport Protocol—the ubiquitous "http" part of a Web address. But HTTP was designed to move HTML data.

For an Internet connection to transport XML instead of HTML for a Web service, a new mechanism was needed to allow XML data to piggyback on HTTP messages, the means by which Web sites receive commands from the keyboards of surfers and transmit data back for display. That mechanism is a new standard, Simple Object Access Protocol (SOAP), developed by independent programmers in conjunction with researchers at Microsoft Corp., in Redmond, Wash.

The SOAP standard was invented to "overlay" XML over HTML in a commonly understood way. SOAP acts as a generic wrapper for transmitting bits of data. It's a kind of envelope that doesn't know what's inside but is recognized and accepted by Web browsers and servers.

Together, XML and SOAP give Web service applications unparalleled interoperability. In fact, in principle, a Web service can be written to use databases that the application developer didn't even know existed. That's a tremendously useful feature for certain applications—for example, older mainframe applications that can be given a new lease on life through a Web services interface. Imagine a freight forwarder writing an application that accessed a musty U.S. Customs database of commodities, so that an importer could figure out his or her customs duties before making a shipment.

For such an application to work, Web sites have to be able to announce to the service that they contain data—such as clearinghouse information, commodities listings, or an airline schedule—that might be useful to it. So another specification was developed: Universal Discovery, Description, and Integration (UDDI).

Basically, UDDI lets Web services look for databases in the same way that Google lets humans look for Web pages. One way that's done is through UDDI registries, a Yellow Pageslike directory in which companies list their businesses and the Web-related services they provide. IBM and Microsoft in the United States and Germany's SAP are among the companies that maintain UDDI registries.

Using a search engine, of course, is sometimes a hit-or-miss proposition—you try different Web pages until you find one that has the information you need. That doesn't work so well without a human to make those judgments. Thus, one more standard had to be invented: the Web Services Description Language, or WSDL.

This standard allows a machine to figure out on its own just what's at a site once it's been identified. A program accessing a Web service retrieves a WSDL description from the service. The description itself is specially formatted XML data telling the prospective user the procedures it can call and a little bit about them. UDDI and WSDL are a magnet that works much better than a human when it comes to finding a needle in the haystack that is the Web [see table, "

ENHANCING THE TRADITIONAL WEB

Web services applications mimic traditional Web activities. Here are four layers of traditional activities and their Web services counterparts.

FUNCTION	TRADITIONAL WEB ACTIVITIES	WEB SERVICES
Find a Web site	Search engine	UDDI
Site description	Search engine site descrip-	WSDL
Transport protocol	tion	SOAP
Data format	HTTP	XML

FELIX SOCKWELL

"

All these new protocols, SOAP in particular, took years to develop. Indeed, they're still works in progress, in part because contributing companies want to receive patent royalties or just don't want a competitor to control a standard. Those same concerns sabotaged two earlier transport mechanisms, one from the Unix world and one invented by Microsoft.

These attempts failed because they didn't provide the freedom Web interactions need—the casualness of vehicles and highways, where anything from a motorcycle to an 18-wheeled truck can travel on any road and go almost anywhere. In other words, any Web client program running on a server, a PC, or even a PDA or Web servicesenabled cellphone, can establish communication with any Web service on the fly. This property is known as delayed binding. Traditional applications, on the other hand, will stop working when a change is made in one component, such as adding one more parameter to a procedure, if this change is not propagated to the rest of the software system.

The practical impact of delayed binding is enormous. Because of WSDL, a program calling a Web service can check the configuration of the Web service as the program runs, allowing the calling program to adjust for any changes that may have occurred in the Web service. This lets programmers separately develop and test the different components of an application, which will continue to run correctly even if one of its constituent modules is upgraded.

The ability to change one part of an application without having to revalidate the whole system radically reduces development costs. Because of these savings, we'll soon see Web services even in complex business operations, such as the processing of insurance quotes or mortgage loans, where different parts of the process can reside in different organizations or companies and run in a variety of computer architectures. The loose coupling and delayed binding properties of Web services will let companies gradually replace older software and interfaces without the disruption of massive software upgrades.

Of course, benefits like these come at a price. There are extra run-time checks, and the text-based data used in XML makes it inefficient. So applications using Web services are several times slower than applications using binary data. In addition, sending plain-text XML across the open Internet makes it vulnerable to security breaches. A number of research efforts are addressing these shortcomings.

However immature, Web services are providing tremendous benefits today. When you add Web services to a classic database, such as a large retailer's inventory, unexpected and delightful applications emerge. Take Amazon.com, one of the largest databases to be opened up to Web services. For almost a decade now, affiliated companies and weekend programmers have been experimenting with it incessantly. Many of the more successful programs were compiled in a recent book, Amazon Hacks (O'Reilly, 2003). Author Paul Bausch says his favorite program involves an Amazon feature known as the "wish list." As you browse Amazon, you can add books to your personal list, a way of not forgetting them. The hack in question makes the wish list viewable on your cellphone. "If I'm in a Borders bookstore, I can answer the question, what was that book I wanted?" Bausch says.

Because of its pervasiveness, the Web is a subject of intensive research, so, of course, there's a next stage after Web services. It's called the semantic Web [see "Weaving a Web of Ideas," *IEEE Spectrum*, September 2002, pp. 6569 Although Web services allow a machine to publish its data, making it available to another machine, the two have to agree on the structure of the data they are publishing. In the semantic Web, this sort of agreement will be largely unnecessary.

For example, an airline and a commuter railroad can publish their respective timetables, but if they don't agree on how timetables are described, programmers who want to develop, say, an automated travel agent application have to manually set up a translation from one to the other. The semantic Web is an attempt to create frameworks that allow the airline and the railroad not just to publish their data but also to offer information about the structure of their data, so that the process of translation can be automated.

But most of the semantic Web's benefits won't be seen for some time; Web services are here today. The Web is a ubiquitous, almost transcendent phenomenon, yet its future has just begun to be tapped. It will connect almost every island of data, software, and device on the planet. The conceptual work has been done; it's time now for the heavy lifting of application development. It's time for the second Web.

There are many good sites for learning more about SOAP. One of the best is SOAP News, maintained by Dave Winer, who was instrumental in developing the protocol.

The World Wide Web Consortium maintains the XML standard and several others related to Web services. See http://www.w3.org/2002/ws/.