Web Services, Part I:
Back to the Future of Software

By Kevin Werbach

As Al awoke one morning from an uneasy dream known as the dotcom bubble, he found the computer industry transformed . . . yet strangely recognizable. (He was relieved he hadn’t turned into a giant insect!) The now-ubiquitous Internet seemed to have had little effect on the landscape. Once again, platform battles raged, as big vendors fought for the allegiance of developers and enterprises. IT organizations faced the familiar central challenge of making systems scale and interoperate.

Many of the players and technologies had changed. IBM, once the strongest advocate of single-vendor centralization, now was a leader in the charge for loose coupling and open standards. People used terms like Web services and business webs where before they had talked of objects, middleware and value-added networks. Yet the goal was the same: to move from monolithic systems to distributed components that made up for their lack of power with superior flexibility, robustness and communications skills.

We’re now living in Al’s world. As growth rates, valuations and business models regress to the mean, young developers’ thoughts turn to age-old challenges. You can see the glass as half-empty or half-full. On the one hand, for all the billions spent on Internet technologies, it’s still hard to connect your order-processing system to my supply-chain system to someone else’s tariff calculator. Below the hood of most large Websites and e-businesses you’ll find plenty of Scotch tape and Krazy Glue.

On the other hand, core Internet technologies and standards have been universally adopted, creating a new foundation for boundary-
spanning services. There is surprisingly widespread agreement on the mechanisms for moving forward from here. It was naïve to think we’d be at the finish line by now, but we may be at the starting line of a more important race: the race to transform software as dramatically as the Web changed publishing and information access.

We'll cover these developments in two parts. Below we describe the Web services concept and the companies building development platforms around this model. Next month, we’ll look deeper into the new world that Web services promise, and will consider the impacts on industry structure and technology.

What’s a Web Service?

Web services are software objects that can be assembled over the Internet using standard protocols to perform functions or execute business processes. They fill in the white space between applications, systems and companies with simple messaging, description, discovery and management protocols and other mechanisms. Anything can be exposed as a Web service, from trivial Fahrenheit-to-Celsius temperature converters to full enterprise applications. To
use what is fast becoming a canonical example, the provider of an online portfolio application could call a stock-quote service over the Internet using an open format, rather than hard-coding in the feed. It could use the same mechanism to add other services or swap out quote providers, because everything uses the same protocols.

Web services seek to realize a widely shared vision for the future of software: distributed, componentized, standardized, open, scalable, and Internet-centric. Ironically, such software can be delivered and paid for as fluid streams of services because it is built from self-contained modules and objects. Then again, quantum mechanics teaches us that the continuous world we see around us merely approximates the discrete bundles of energy that make up the universe.

“We have this sea of connectivity and all these islands sitting in it,” says Charles Fitzgerald, director of business development for Microsoft .Net and a leading Web services strategist. Fitzgerald labels Web services “an Internet-native integration methodology,” and rhetorically poses the challenge: “How do we really start to solve the problem of computers talking to computers?”

A brand new day?
At a general level, Web services are hardly a new idea. “Everybody who ever thought about distributed computing understood that this is how distributed computing will work in the future,” observes Idoox ceo Roman Stanek (see below). [Disclosure: Esther Dyson is an investor in Idoox.] What has changed is the technical and business context, compared to that of earlier component efforts such as Microsoft’s distributed component object model (DCOM), the Unix-oriented common object request broker architecture (CORBA) or Sun/Java’s remote method invocation (RMI).

The breadth of support for the model is striking, especially at this stage. Microsoft has done more than anyone to fuel interest in Web services by promoting the idea in connection with its .Net platform for Internet-centric applications. Other industry heavyweights including Sun, IBM, BEA, Oracle and H P (see box, page 9), as well as a bevy of startups, all agree that Web services are the next big thing.

In one way or another, these companies hope to do for applications what the Web did for publishing: reduce costs, catalyze innovation and dramatically lower the bar for anyone to create new offerings. (We first heard this analogy from Clay Shirky of the Accelerator Group.) Instead of mostly static links between data or content, Web services promise active links (transactions) between functions.
Beyond the near-term integration benefits, promoters of Web services envision a world where every company connects to every other, and systems are assembled and optimized on the fly to meet dynamic business requirements. In broad brush, it’s the universal syndication vision we described in the past (See Release 1.0, July/August 1999). Syndication is an over-arching concept for business relationships in a distributed, networked world, though so far it has applied mostly to content. Web services are a specific set of technologies and standards to implement this vision, roughly corresponding to the “application syndication” category in our earlier framework.

At your service?
The term “Web services” is easy to pick apart. Web services go beyond the Web, which is a publishing medium rather than an application environment. And just what is a service, as distinct from a Web application (See Release 1.0, January 2000)?

A service is something that does something you want, generally across some dimension of time or space. Internet service providers offer ongoing Internet connectivity for a monthly fee. Customer service centers answer calls and provide assistance (sometimes). Unlike applications, which are self-contained, services tie things together and extend beyond their own boundaries. And they can interface to other services, not just to humans.

There are plenty of services on the Internet today. ICQ and other instant messaging clients allow users to communicate with one another. MapQuest takes an address and returns a map, with businesses or other features marked if you desire. Akamai adds distributed caching to the Web’s centralized client-server content-delivery model. Microsoft’s Windows Update site identifies and serves up software updates that your machine needs. DoubleClick turns banner ads from an integral part of a Web page to a component dynamically targeted every time a page loads.

All of these are Web services in the general sense, but they are missing an important element. The application programming interfaces (APIs) through which other software talks to these services are defined by the service owners. Those APIs may be well-designed and well-documented, but anyone who wants to use them will have to learn the particular syntax of each. That limits adoption. Such brittle one-to-one connections are similar to the important but expensive electronic data interchange (EDI) links between enterprises.
Web services may involve many standard protocols. Things are moving so quickly that press releases and even implementations by companies often precede completion of the standardization process. And it’s easy to get a misleading picture from the oft-used Web services stack diagram we reproduce on page 6 as Figure 1.

No one service is likely to invoke all the standards listed below; the only piece that’s truly essential is XML. Web services run the gamut from small, ad-hoc projects — such as Evan Williams’ XML-RPC interface to his Blogger Weblog-creation tool (see Release 1.0, May 2001) — to the largest enterprise business-to-business integrations. Different protocols are appropriate for different kinds of implementations. In the list that follows, everything below SOAP is being promoted most actively by those focused on integration among large businesses and trading partners. We’ll trace the different visions for Web services in more detail next month.

The list below is in roughly descending order of importance and current industry adoption.

XML (Extensible Markup Language) — World Wide Web Consortium (W3C) standard for representing any kind of structured information. Derived from the more-complex and therefore less widely adopted standard generalized markup language (SGML).

XML is being universally accepted and widely supported because of its simplicity and flexibility. It is the foundation of all Web services, and the remaining entries on this list are all XML-based protocols. However, XML addresses only the base level of interoperability between systems. By analogy, you can connect to France by phone, but that doesn’t mean the French can understand you.

SOAP (Simple Object Access Protocol) — A mechanism for exchanging structured messages between distributed Web services. If XML is the phone system, SOAP is the common language for expressing service requests. Initially developed by Microsoft, Userland and Developmentor, though it has now been endorsed by IBM, Sun and many others, and has been submitted to W3C. Although most business implementations use SOAP, there is also a simpler alternative called XML-RPC created by Userland’s Dave Winer (see Release 1.0, July-August 1999) which some developers prefer.

UDDI (Universal Description, Discovery and Integration) — A protocol for Web-services directories, so that companies can automatically identify remote services, their functions and their providers. Developed by IBM, Microsoft and Ariba and now promoted by a private standards organization with nearly 300 members (see Release 1.0, December 2000). Version 10 was published in draft form in September 2000; version 2.0 was released in June 2001. Work is now underway on version 3.

WSDL (Web Services Description Language) — Explains a particular Web services interface and how the messaging protocol (SOAP) can connect to it. In other words, companies can use WSDL to write detailed structured descriptions of Web services residing in UDDI directories or elsewhere. Submitted in March 2001 to W3C by Ariba, IBM and Microsoft, the same trio behind UDDI.

WSFL (Web Services Flow Language) and XLANG (Extensible Language) — Two proposals, likely to be combined somehow, for orchestrating Web services. WSFL was created by IBM, while XLANG was developed by Microsoft. These protocols cover the temporal dimension of Web services: which components need to be invoked in what order, under what conditions, with what latency requirements and with what expectations or error-correction mechanisms. These are crucial issues for robust end-to-end services, but the protocols are relatively unsettled.

EbXML (e-Business Extensible Markup Language) — Developed by the Organization for the Advancement of Structured Information (OASIS), a UN-affiliated group, for business-to-business (B2B) integration. Confusingly, EbXML incorporates features of other standards listed above such as UDDI and WSDL. More confusingly, several of its major backers have also endorsed these other protocols. OASIS is also managing the standards effort for business transaction protocol (BTP), designed to manage complex B2B transactions using Web services.

DSML (Directory Services Markup Language) — Protocol promoted by Bowstreet, a Web services integration company we cover below, and endorsed by companies such as Microsoft, Oracle, Sun and IBM. DSML allows companies to access remote directories to exchange business-to-business commerce information.
The XML revolution

The core differentiator of today's Web services can be expressed in three letters: X-M-L. XML is the extensible markup language (see release 1.0, May 1998), the common protocol Web services use to expose their interfaces.

As technologies go, XML is quite simple. It's just a format to identify and tag fields for easy data exchange. Other protocols could have been picked. The important fact is that something is now universally accepted for communication between (and within) Internet-based applications and services. Previous component-software efforts forced developers to choose between proprietary alternatives. They were also too heavyweight and desktop-centric for today's Internet-oriented world.

Engineers use the term "loose coupling" to describe integrations with limited cross-dependencies. On the Web, for example, if a page you link to moves, you get a 404 Not Found error and possibly a referrer page. The Web doesn't break; it just tells you there's an exception. Previous object models didn't go far enough in that direction. Microsoft's Fitzgerald, while pointing out the $1-billion third-party market for components based on Microsoft's COM, acknowledges that, "Any of those tightly coupled object models don't work over the Internet. What you need is a model where potentially any asset connected to the Internet is a programmable building block." That's the promise of XML-based Web services.

XML also has the virtues of being simple and human-readable. In this way, it resembles HTML, which allowed millions of individuals and small businesses to serve up formatted content to a global audience. HTML's radical simplicity came at a cost: It offered only a limited set of attributes designed for publishing static Web pages. XML removes that limitation, allowing formats (known as schemas) to be defined as needed. The great thing about XML for Web services is that its attributes are self-defining. If Sam builds a stock-quote application and defines fields for price and volume, he can enclose them with <price></price> and <volume></volume> tags, making it easy for Ethel (or a system she builds) to parse what he was trying to do.

Of course, nothing's that simple. Making raw XML useful requires a variety of other standards and schemas implementing those standards so that, for example, Ethel can...
match her format for a price with Sam’s. (Otherwise, Ethel may send the price of an IBM PC rather than of IBM’s stock.) And XML itself is only one of a suite of standard protocols that have been proposed for Web services (see box, page 6). Having a common syntax or wire format, which XML represents, is the crucial step for Web services to take off. Standardization higher up the stack should further increase the adoption rate and power of the model.

On the other hand, none of the other standards is as widely supported as XML (though SOAP is coming close). And no standard is perfect. UDDI, for example, was designed with the needs of large business-to-business trading partners in mind, and thus may not be the best mechanism for other kinds of Web services.

A real-world example
For all the talk and excitement around Web services, actual implementations are few and far between. Web-services directory XMethods (see resource section) lists dozens of examples of SOAP-based services, but most of these are experimental. Several of the vendors below point to marquee customers, but there are few case studies of completed implementations beyond internal integration projects.

The integration between Southwest Airlines and Dollar Rent A Car is one real-world example. Southwest tries to drive online reservations through its own Website rather than third-party sites such as Travelocity. It was interested in offering customers the ability to make rental car reservations through Dollar when they book flights on its site. However, Southwest runs Solaris and CORBA, while Dollar uses Windows and a VAX-based back-end reservation management system called Quick Keys.

Systems integrators quoted millions of dollars and months of development time for the project. Instead, Dollar built a SOAP-based messaging interface between the two systems, effectively turning its reservation engine into a Web service (see figure 2). The prototype took just two weeks, and the full implementation about two months. The relationship now generates $2 million in incremental annual revenue for Dollar.
Cost savings are only one benefit. Explains Charles Fitzgerald of Microsoft, which provided the Web-services tools to Dollar: “When you talk to them, the thing they really go nuts about is that they had a business revelation. There are a whole bunch of things they can now go do if integration is a lot cheaper and faster than it has been historically.” Once the SOAP-processing infrastructure is in place, it’s straightforward to add additional linkages at significantly less cost than the first integration.

Services With a Smile

The idea of Web services encompasses many kinds of companies. Further confusing the issue, any vendor that builds some XML or SOAP support into its products – a relatively trivial effort – can claim to offer “Web services” functionality.

At this stage of the market, there’s a relatively clean break between the large platform players such as IBM and Microsoft (see box, below) and the mostly new companies seeking to carve out higher-level niches. The platform vendors are all adding the ability to create native Web services to their development tools, such as Microsoft’s Visual Studio.Net, and they are building connectors to encapsulate and expose existing components with Web services protocols. The companies described below are creating tools and services that take advantage of those platforms, applying them to business challenges. Others, which we will discuss next month, look to offer complementary services and functions.

Before long, consolidation is bound to happen. In most cases, the platform vendors will acquire startups to make their offerings more comprehensive. That process has already begun with BEA’s $30-million acquisition of Crossgain, a small company led by many of the original architects of Microsoft’s XML and Web services efforts. Crossgain was a unique case, though – it was under heavy pressure from Microsoft for hiring away employees still under non-compete agreements.

Web-services pioneers

Web services are an approach, not a market. Many vendors use Java as their core platform, but hardly any call themselves “Java companies.” Likewise, no provider describes itself as an Internet company; that is no longer a distinction.

The Web services companies we discuss below are not quite direct competitors, though there are areas of overlap, and they could compete in the future. Bowstreet
Web services are at the center of the next major software platform battle. Microsoft’s .Net has forced others to decide whether to play within Microsoft’s ecosystem—as companies such as eBay, Groove and Ximian have chosen to do, at least in part—or to offer a competing framework. Even those who choose the latter course agree on the basic approach of networked components communicating through XML, SOAP, UDDI and WSDL.

Microsoft .Net
Microsoft first bet on XML as the software integration format for the Internet era around 1997. The company deserves credit for aligning itself (and others) around the Web-services model. Skepticism has persisted about Microsoft’s devotion to an open standard, but the company’s early and intensive commitment to XML and SOAP is paying off. It is ahead of most vendors in the level of Web-services support in its products.

Still, there remains a great deal of confusion about .Net, the overarching strategy that Microsoft announced last year. .Net includes Web-services development tools; support for SOAP and related protocols in Microsoft’s server products and client software such as Windows XP; Hailstorm, a “platform service” for identity-based Web-services; and other elements, including the C# programming language and common-language runtime, unrelated to Web services. Some see Hailstorm as an attempt to create a new monopoly over the Internet; others simply have difficulty sorting out the pieces and how they relate to the grand vision Microsoft has articulated.

Sun Open Net Environment (ONE)
Sun argues, not without justification, that the basic vision of Web services is the “network is the computer” mantra it has followed for two decades. J ava is a network-savvy cross-platform environment, and it was a Sun engineer, J on Bosak, who led initial development of XML. In February, Sun announced its Web-services strategy, Sun ONE, incorporating these building blocks. In many ways, therefore, Sun is the company that has done the most to build the foundations for a Web-services world.

Nonetheless, Sun’s current commitment to Web services seems half-hearted. Sun generates revenue licensing J ava, which is open as development environments go but is not a free, independent standard like SOAP or XML. With Microsoft using .Net and its C# programming language as a wedge to wean developers off J ava, Sun is in a difficult position. Interestingly, BEA, which has become one of the largest and most prominent champions of Sun’s J 2EE platform, has been more aggressive in adopting and trumpeting Web services.

IBM Dynamic E-Business
Given IBM’s Gerstner-era focus on solutions rather than hardware and software, it’s not surprising that IBM treats Web services primarily as a middleware methodology. The company has an extraordinary technology portfolio and an army of developers and consultants; Web services provides a mechanism to bring together those resources to create solutions for customers.

IBM was the first major vendor after Microsoft to adopt SOAP, giving the protocol a significant credibility boost, and it has been a major player in UDDI and other standards. The company is promoting Web services as part of its WebSphere e-business platform.

HP Web Services Platform
HP was talking actively about distributed network-based services before any other large vendor thanks to its E-Speak initiative, which we covered in our essay on innovation (SEE RELEASE 1.0, JANUARY 2000). Despite this, HP largely dropped off the Web-services radar, participating actively in standards groups but not taking a visible role in the SOAP-oriented world we’ve described.

The company can, however, point to several E-Speak implementations, including electronic-components supply-chain automation company SpinCircuit.com, as well as its own global support logistics unit that provides a marketplace for HP printer replacement parts. Recently, HP announced plans to release its Web services platform, the next generation of the E-Speak software, in November.

Oracle Web Services Framework
Oracle is incorporating Web-services standards into its flagship relational database software, and has announced a Web service brokering engine and management tools. However, the company has so far not been very aggressive in promoting these offerings. In recent years Oracle has been a strong advocate of a single-vendor suite of enterprise applications, in contrast to the best-of-breed approach of its competitors. This is the exact opposite of the mix-and-match ideal of Web services, so Oracle may have difficulty embracing the component model.
has the grandest vision, pushing the idea that companies can transform themselves by applying mass-customization techniques to their applications. Idoox is at the other end of the spectrum, telling a familiar developer story transposed into a new technical context. The Eon Company and Avinon seek the age-old goals of improving productivity and lowering barriers for non-technical business analysts to manage processes. Avinon takes more of a workflow/process approach, while Eon offers a multi-layered application-development framework.

There are many other companies we could include but haven’t, because they use Web services more as a means to achieve other goals. Crossweave, iSpheres, Cacheon and Asera integrate internal and external systems into virtual composite applications. Others, including Kenamea, Bang Networks and KnowNow, enable real-time interactions over the unreliable Internet. SlamDunk Networks and TransactPlus provide secure reliable messaging to support transactions across the Net. The categories are approximate, and everyone is a moving target. We plan to look at some of these startups in a future issue.

Bowstreet: the application factory
Bowstreet got its start in 1998, when XML was just becoming a standard and well before the term Web services was being used. It has gone through several refinements of its message as it builds out its products, always animated by a core vision. Bowstreet’s big idea is that business software is due for its own industrial revolution, moving from labor-intensive, single-purpose crafts to reconfigurable services fabricated through automated processes.

Bowstreet co-founder and chairman Frank Moss was previously CEO of Tivoli. After selling the company to IBM, he began funding or advising a number of startups broadly related to the shift toward Web services. He argues that as business becomes native to the Internet it will move from centralized hubs to distributed “business Webs.” For that shift to happen, he says, “applications as we’ve known and loved them for 40 years are going away.” Relying on programmers to hard-code applications won’t scale to the number of dynamic relationships required. “You’ve got to look at a model that takes a lot out of the hands of programmers and gives it to business people and ultimately to end-users,” Moss explains. Web services are the means to that end.
Moss sees most Web-services vendors focused on the initial middleware challenges of messaging and exposing XML interfaces, when the real transformative step is to take those Web services and assemble them on the fly into applications. As CTO Andy Roberts argues, “The middleware folks are all focusing on technology to break the monolithic applications into pieces,” whereas Bowstreet concentrates on putting those pieces back together in novel and flexible ways. As the number of components and users increases, the complexity of the process scales exponentially, necessitating an automated development approach to pump out customized applications.

The technology Bowstreet uses to achieve this, parametric modeling, is derived from mechanical process-automation applications such as CAD/CAM. Developers create a high-level generative model that includes tweakable parameters, and the application is assembled dynamically from that model. Change a parameter, and the application automatically reconfigures itself. “What we realized is that there are huge similarities in modeling a large complex system like a jet engine and modeling a Web application,” says Roberts. The reconfigurable elements in mechanical engineering are geometric features such as the size of a hole in a part, while in applications they are things such as user interface, workflow and back-end connections to legacy systems. These aren’t always quantitative, but to the extent they are, the parametric approach pays dividends.

For example, if a company offers three levels of service to different customer categories, it can build one application that presents the right interface and functions to whomever comes in the front door (given supporting identification mechanisms). Enterprise portal tools offer some of this customization capability at the presentation layer by building interfaces from templates, but they have limited ability to manipulate the underlying application logic without inflexible hard coding.

Moss believes composite applications, taking up where portals leave off, will be the first popular application of the Web services methodology. Eventually, this approach will allow companies to link themselves more tightly into dynamic value chains involving partners, suppliers and customers. “We see services being the links which bring together these value chains,” says Moss.

In the client-server revolution, the most highly-valued companies were ultimately the application vendors such as SAP and PeopleSoft rather than the tools vendors such as Powersoft. Moss thinks that will be reversed in the Web services arena. “I think the tools are going to be the drivers of this revolution. The applications can be created at will,” he predicts.
Idoox: everything you need to create Web services

Idoox has been working on Web services for over a year, making it one of the first startups to focus exclusively on this space. Ceo Roman Stanek sold his previous company, Java tools vendor NetBeans, to Sun, and then left to start Idoox.

“What I learned at NetBeans is that if we want to be successful, we not only have to build tools; we have to build infrastructure: something people use at runtime,” says Stanek. Idoox offers a variety of technologies companies can use to create, publish, discover and use Web services. It provides security and transactional capabilities that go beyond simply exposing an interface as a Web service. Where NetBeans addressed the respectable but limited tools market, Idoox will give away tools for free to build demand for its software.

The Idoox Web application and services platform (WASP) includes a secure, high-performance SOAP runtime environment that integrates with the J2EE platform. It also offers a development framework based on WSDL (see box, page 5) and tools that can plug into popular Java development environments including NetBeans, Sun Forte and Borland JBuilder.

“When I say we are in the Web-services industry, people think we compete with Razorfish,” Stanek notes. The current funding climate exacerbates the problem, because the Web-services model isn’t yet sufficiently developed for predictable revenue streams. Stanek says there’s a natural threshold any new infrastructure technology must cross before it becomes useful. Otherwise, “it’s like a car with three wheels.”

UDDI is one area of Web services that is still developing. The standard was originally announced last September, but Stanek thinks it will take until the third version (now in early stages of development) before the platform settles down to the point where companies are truly comfortable building on it. Idoox is developing a distributed UDDI registry, which will be simpler for companies to deploy than the centralized registries envisioned by the large companies behind the UDDI standard.

The company’s Enterprise UDDI offering supports the full specification but adds extensions to meet security and reliability requirements of enterprises. For example, policy management allows requests to be redirected when Web services are unavailable. Explains Anne Thomas Manes, who recently left Sun to become the cto of Idoox: “The public UDDI registry [operated by Microsoft and IBM] is equivalent to the yellow pages. How do you know that a business advertised in the yellow pages is
legitimate? But private registries are very different. A private registry is managed and controlled by a private business or community. This private community can establish whatever security and reliability controls it wants.”

Stanek thinks the biggest selling point of Web services is that, “It brings together what people use externally and what they use internally.” In other words, it brings the benefits of the Web to the tightly controlled realm of internal IT. Where before departmental workgroups may have used quick-and-dirty Internet technologies but enterprise systems still depended on heavyweight proprietary interfaces, now everything can ride on the same XML layer and can offer full transparency to anyone authorized. “You actually encapsulate your centralized computing to Web services, and you let people on the edges invoke it,” Stanek says. Unlike client-server, which also sought to unlock centralized data, Web services doesn’t require anything new at the edge of the network.

“We would like to be a BEA Weblogic of Web services,” says Stanek. Why won’t BEA be the BEA of Web services? we ask. “We see Web services as an opportunity. BEA and Sun have well-functioning J2EE franchises, and they don’t want people to start asking whether to use J2EE or Web services.”

Avinon: automation through scenarios
Avinon helps companies create dynamic business services based on “scenarios,” which are high-level abstractions of processes and functions. Its ceo, Kamran Kheirolomoom, spent 15 years in the workflow trenches as the founder of ViewStar, which was merged into call-center software vendor Mosaix. (For those who like to follow the bouncing ball, Mosaix was later acquired by Lucent, and then spun off as part of Avaya.) Kheirolomoom believes his long experience modeling business processes gives Avinon a leg up in figuring out how to use Web services to empower business users.

“The problems that Web services try to address are real problems that have been there for years,” says Kheirolomoom. “What’s new is you’ve got a unifying layer called the Web for delivering services.” Although Avinon makes it possible to tie together different applications, back-end integration isn’t its focus. Avinon concentrates on the processes and interactions between an enterprise and its customers or partners. Its tools are designed to allow companies to automate activities without having to engage in extensive program-
If you want to expose those enterprise assets to enable these interactive business services, use Web services as the glue layer for making those assets visible and accessible to your front-end application," says Kheirolomoom.

Avinon provides a drag-and-drop environment in which users can snap together Web services and other functional building blocks to create applications. Users can then publish the application to a Website, portal, partner site or rich email, with the look-and-feel dynamically altered to match the environment. If a process changes, business analysts can go in and alter application logic in the same interface. For example, an insurance carrier subject to a changing regulation could drop in or edit some building blocks, then simply re-publish an online policy quote service.

Avinon vp of product marketing David Ruiz, a co-founder of ViewStar with Kheirolomoom, says that compared to traditional workflow, which focused on automating back-office processes, Avinon focuses on automating the market-facing processes within an enterprise. “It includes automation of the actual interactive experience as part of the modeling environment,” he explains.

Avinon sees itself occupying the high ground in the Web-services landscape, focusing on the orchestration of business activities of highest value to customers rather than building enabling platforms or connecting systems at the back end. “Most of the application framework vendors provide tools for IT people to do what they do best. We’re trying to establish a collaborative model where online solutions are co-created by IT and businesspeople, each with access to the functionality best suited to their skill set, experience and role on the team,” says Ruiz.

Kheirolomoom argues that enterprises will initially adopt Web services as a better way to do internal systems integration, and then will create new applications and services for their own use and for customers. Next they will extend those Web services out to existing trading partners, who already represent trusted relationships. It will be 18 to 24 months, Kheirolomoom believes, before companies start using Web services extensively to do business with entities they don’t already know. (At that point directories and authentication will become much more important.)

In the end, Kheirolomoom agrees that the business impact of Web services will be as great as the technology shift. Software as a service implies thinking about all business functions as services, which can be swapped and aggregated as needed. That’s when businesspeople will really start to get excited about the concept.
The Eon Company: packaged infrastructure

“You two should talk,” said Invesco managing director Alessandro Piol, motioning to the person next to him as we sipped mixed drinks on the rooftop patio of a Manhattan restaurant. Alessandro is a hard man to ignore, so when Mitch Sonies, CEO of the Eon Company (and a fellow recovering lawyer) got in touch the next day, we agreed to a meeting. We’re glad we did.

The Eon Company makes what it calls “packaged infrastructure” for e-business applications. It positions itself between general-purpose platforms such as application servers (BEA Weblogic, IBM WebSphere, etc.) and packaged applications such as Siebel or Broadvision. If a packaged app doesn’t do what an enterprise needs, the usual alternatives are customization, which is usually expensive and difficult, or creating something from scratch on top of an application server... which is time-consuming, even more difficult and often equally expensive.

Eon’s answer, Eonworx, addresses these problems through a heavily layered architecture, integrated with common e-business application building blocks. The bottom layer is a relational database such as Oracle or Microsoft SQL Server, but on top Eonworx includes a series of abstractions. At the highest layer, non-programmers can design applications by rearranging business processes through a visual interface. The system automatically transforms those instructions into XML and from there into lower-level components, connectors and ultimately database queries.

A key link to Web services is that all the layers talk XML both internally and externally. Eonworx provides native support for standards such as SOAP, and allows layers and services to be swapped out as desired. In other words, Eonworx isn’t designed primarily to create Web services, but it gains its flexibility and power by leveraging the Web-services approach. Every Eonworx component is called by passing a URL request and getting some content to render in response, meaning no applets, active server pages, or other client-side logic. “This is one reason why physical location or client device is completely irrelevant all the way down to the most granular component level, from a protocol standpoint,” explains Sonies. “Eonworx functionality can be broken up and spread across the Web as most efficient.”

He continues: “When we started, Web services didn’t really exist, though we thought it was pretty obvious this was how you wanted to build everything. We’ve been focused on why you would want to do this, which was primarily to create efficiencies.
in the way you acquire and maintain things. Most of the big companies we talk to are more immediately interested in that and in the ROI than in Web services per se.”

Sonies admits that even at the high levels in the stack some technical proficiency is required, but he pegs it at the level of a junior consultant at a firm such as Accenture. “Our business app developers do have to know basic Javascript, XML and UML, which is far from writing object-oriented procedural code in Java (it’s actually a lot easier than Visual Basic), and they need a few weeks of training,” he says.

Sonies emphasizes Eonworx’s comprehensiveness, incorporating a development environment, workflow, messaging, data integration, internationalization, personalization and content storage, rather than simply offering a mechanism to link up services. The company’s in-house developers are also creating specialized high-level service components to cover the most common functions in popular application domains such as customer relationship management and supply chain event management. Eon’s first production customer, Telia, is now going live, and beta customers include Fiat. Eonworx will be officially launched in the first quarter of 2002, and will be offered as either a hosted service or licensed software. Eon also is signing services providers and systems integrators to distribute and resell Eonworx.

An Evolutionary Revolution

Nothing new under the sun
As we’ve stated, Web services draw on old concepts. In fact, it’s remarkable how many other ideas and trends Web services resemble:

Web services allow different applications to share data and logic... but so do enterprise application integration (EAI) tools from vendors such as Vitria (see release 1.0, March 1999), CrossWorlds and WebMethods.

Web services make it possible to build applications out of re-usable components that can invoke one another... but so did Microsoft’s object linking and embedding (OLE) and Apple’s OpenDoc in the early 1990s (see release 1.0, May 1994).

Those components can be distributed over the Internet... but so can objects built with DCOM and CORBA.
Web services allow software to be delivered as a service, rather than as a stand-alone application. . . but so do application service providers such as Corio, Salesforce.com and Jamcracker (SEE RELEASE 1.0, SEPTEMBER 1998).

Those services can plug into and enhance applications. . . but so can “meta services” such as Akamai’s content-delivery network (SEE RELEASE 1.0, DECEMBER 1999).

Web services push application logic out from central services to the edges of the network. . . but so do peer-to-peer (P2P) infrastructure platforms and tools such as Groove (SEE RELEASE 1.0, MARCH 2001 AND DECEMBER 2000).

Web services can dynamically organize information around users, hiding the distinction between local and remote content. . . but so do “data soup” aggregators and P2P applications such as Napster (SEE RELEASE 1.0, APRIL 2000).

Web services can pull together information from several sources, giving users a unified view into content and applications. . . but so do enterprise portals (SEE RELEASE 1.0, FEBRUARY 1999).

Web services allow applications to be split up and recombined across the network in optimal arrangements. . . but so does application syndication (SEE RELEASE 1.0, JULY/AUGUST 1999).

That’s just a sampling. With more time to go through the archives, we could have added references in the list above to numerous earlier issues of RELEASE 1.0.

The point is not that we’ve duped you by writing about the same topic repeatedly (and are doing so again this month!). There are certain fundamental concepts that reappear over and over again in the history of computing, taking on different forms to fit the particular technological context and specific needs. In some cases, similar ideas have different effects when the world around them changes. Unix is an old operating system, but Linux gave it a new lease on life thanks to the open-source model and the Internet. Because of today’s environment, Web services may achieve many of the goals described above more easily and across vendors and platforms.

There are also deep trends that manifest themselves in different ways. In this case, the idea is the atomization and re-integration of software, moving from monolithic wholes to piece-parts that can be pulled apart or put back together as needed. Web services are the latest development in this thread, and they won’t be the last.
DIGITAL HARBOR: ANY PORT IN A STORM?

Any technology that TCP/IP co-creator Vint Cerf calls "the most exciting thing I’ve seen since Mosaic," is worth checking out, especially when it has a history in the demanding world of military intelligence. And it's hard to deny the scope of Digital Harbor's vision.

The company offers nothing less than a complete next-generation platform for Internet computing, embodied in more than 10 million lines of code. At first glance Digital Harbor's PIIE (personal interactive information environment) seems to realize the potential of object-oriented computing, in particular of Java. Yet, to continue the nautical metaphor of the company name) it also feels like an effort to boil the ocean, which may founder on the shoals of user familiarity with existing environments.

We describe PIIE separately because it isn't a Web-service platform, though it rides on XML and supports the usual protocols. It's a related branch of the same tree, component software, offering the promise of radical change rather than iterating on familiar platforms.

In PIIE, everything is an object, down to a granular level. That means that, as with Apple's OpenDoc and Microsoft's OLE, any application can call any other application. But PIIE goes deeper. Any content from any application can be integrated or linked to any other application.

Users download a 300K reader application; subsequent application code is downloaded as needed with associated content. PIIE's "console" interfaces shift context on the fly when "appliances" such as maps and calendars are invoked. PIIE ships with two dozen appliances, with others planned. It's easier to show the power of this approach than to describe it, so if you're interested, go to (http://www.dharbor.com/mortgage.html) for a demo.

The Web uses a centralized architecture, which has the advantage of allowing a single relatively dumb piece of software, the browser, to be a standard universal client. The downside is that functions such as drawing a map that require application logic must be done at the server, adding significant latency when the page must be redrawn and sent over the Internet to the user. Java offered transportable code that would allow applets to run locally, but performance and compatibility issues have greatly limited the use of Java on the client side.

PIIE uses its own internal protocol, ATP, which is more efficient than HTTP or other existing Internet protocols for large numbers of small packets. This allows it to handle the huge level of messaging traffic necessary to coordinate so many interacting components, and to minimize network traffic in order to deliver high performance to the end user.

Digital Harbor's challenge will lie in finding killer applications for its platform. As with other powerful, open-ended tools such as Groove (SEE RELEASE 1.0, NOVEMBER 2000 AND JUNE 2001) and Zaplet (SEE RELEASE 1.0, MARCH 2000), PIIE may have great technology, but that won't matter much if it doesn't address real-world needs of paying customers. In fact, the scope of such systems is both their greatest asset and the source of great risk, because it's not obvious just where the killer apps will come from. Given the desire of many companies to find an alternative to Microsoft's .Net platform, however, Digital Harbor could wind up partnering with larger players in a better position to promote its technology.

Digital Harbor is a subsidiary of Eidea Labs, based in Annandale, Virginia. Eidea is a technology development company started in 1997 by CEO Rohit Agarwal, which began the basis of PIIE as a US Defense Department contractor. The company has over 70 employees, primarily developers; it plans to launch officially in October.

Software constituencies
Why has the concept of Web services been so widely adopted?

The software industry has at least four constituencies: developers, vendors, customers and users. This leaves out third-party service providers and channels among other things, the same kinds of arguments would work in a more detailed analysis. Some developers are vendors and some customers are users. For these purposes, the distinction is that developers and users are individuals, while the other terms include organizations (which may have interests different from their members).
Developers (who write the code) like Web services because they can make the process of creating software faster and simpler. They like the idea of re-using components, as well as the promise of reducing thorny integration and scaling challenges. Furthermore, code written by an independent developer may now take advantage of – or be called by – a huge array of existing software.

Vendors (who sell the products and services) love the reliable recurring revenue streams that a service model implies. They want to avoid devoting unnecessary resources to integration, both internally and with other applications. Finally, they like the idea that they can specialize and draw on resources elsewhere.

Customers (who write the checks) want software to be cheaper and more effective, two key promises of the Web services approach. The idea that every change no longer requires ripping out legacy systems or paying for systems-integration jobs is quite appealing. Of course, service-oriented pricing can either raise or lower costs, as shown by Microsoft’s proposed new licensing terms for Office.

Users (who do things with the software) want power and ease-of-use to enhance their productivity. Web services offer them Swiss Army Knives where before they were stuck with forks and spoons. Any application now can take advantage of the richness of features and functions available through the network.

Thus, Web services address the pressing needs of each group. Of course, nothing is ever that simple. As we’ve shown in this issue, the story is more nuanced and the ending less clear than it first appears.

Parlez-vous software?
There’s another important reason for the rapid acceptance of Web services. A distributed approach to software is simply essential to deal with the spiraling complexity of the Internet era.

Large Websites such as eBay and Hotmail have experienced much-publicized difficulties keeping their systems operating reliably. When Wall Street and venture capitalists were willing to fund anything with .com in its name, these limitations could be papered over. There were ways to throw more servers and engineers at the problem. After all, everyone knew the Net was wild, woolly and unreliable. That’s not good enough anymore. Even if it were today, it won’t be tomorrow, because the network continues to grow. Approaches that can’t gracefully scale will fail.
Software is like language: an aggregation of lower level units into meaningful larger structures. In language, sentences are made up of words and letters— which take on meaning only in the context around them. A fluent listener knows where the words start and stop, but someone unfamiliar with the language will hear only a meaningless stream of sound. Similarly, binary source code looks like an undifferentiated stream of ones and zeros to a human, but once compiled it turns into comprehensible (to some!) units. Those units, in the form of modules or libraries, interact with one another to form the applications or operating systems users experience.

The metaphor goes further. The power of language, as linguist Noam Chomsky explained, lies in its combinatorial structure. A few simple rules of morphology and syntax allow a small number of letters to make a large number of words, and those words to make an infinite number of sentences, without exceeding the brain's storage and processing capacity. This works because language has strong standards. Generally, a verb is a verb and a prepositional clause is a prepositional clause.

Standards are at least as common in software, but they are much less universal. Some pieces of code work well with other pieces of code, but that's the exception rather than the rule. The problem isn’t just the variety of languages, such as COBOL, Visual Basic, C++ and Java; it's the different ways information is represented. Even if my application generates a purchase order and yours takes a purchase order as input, both in ASCII format, there’s no guarantee the two systems will be compatible. Mine may output <vendor ID, SKU, price, quantity> and yours may expect {itemNumber, amount}. It takes a human to understand how two interfaces relate.

The result is a tremendous amount of complexity and redundancy. Each vendor must build complete systems, either by developing all the components in-house or by ensuring that its components integrate with those on the outside. It’s an imperfect process, even at the level of a single application. Across the Net and across organizational boundaries, it’s a mess.

The long haul
Web-services offerings represent a concerted effort to untangle this mess. Yet they have a long way to go. The bumpy history of previous component architectures should be a warning to anyone who thinks the world will change overnight. To offer one proxy, we’re roughly where Java was in 1995: a thrilling concept with strong endorse-
ment, but not yet a fully realized business ecosystem. It took several years until, with J2EE, Sun and its partners put all the pieces together. Unlike Java, the Web-services idea isn’t centered on any one company, which makes it potentially more significant, but also means coordination will be more difficult.

Even under optimistic scenarios, Web services adoption will be a many-year process. Many of the less-obvious but critical elements of the ecosystem – naming standards, security, business models, etc. – have yet to be developed or standardized. More to the point, there are many vendors but few customers, and many tools to develop Web services but few examples of commercially available services. We’ll consider some of these topics in more detail next month.

If nothing else, the foment around Web services is reassurance that, from a long-term perspective, things are still moving forward in the computing world despite the current uncertainty and economic difficulty. ■ R 1.0
# Calendar of High-Tech Events

2001

Note: Some events may be postponed or cancelled due to travel difficulties in the wake of the September 11 terrorist attacks.

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCTOBER 1-5</td>
<td>FALL INTERNET WORLD - New York, NY. The mother of all Internet trade shows makes its east coast stop. To register, call 1 (800) 632-5537; fax, 1 (203) 559-2814. <a href="http://www.internetworld.com">www.internetworld.com</a></td>
<td>X</td>
</tr>
<tr>
<td>OCTOBER 3-6</td>
<td>IEEE CONFERENCE FOR STANDARDIZATION AND INNOVATION IN INFORMATION TECHNOLOGY - Boulder, CO. Provides the opportunity for users, developers and experts to discuss standards and emerging information technologies, economics, law, society and culture. For more information, please go to <a href="http://www.siit2001.org">www.siit2001.org</a>.</td>
<td></td>
</tr>
<tr>
<td>OCTOBER 8-10</td>
<td>NEXT-GENERATION WEB SERVICES - San Francisco, CA. Infoworld's three-day event on the future of Web services. Speakers include Federal Trade Commission’s Sheila Anthony and BEA’s Bill Coleman. <a href="http://www.eventreg.com/nextgen/nextgen_1.htm">www.eventreg.com/nextgen/nextgen_1.htm</a></td>
<td></td>
</tr>
<tr>
<td>OCTOBER 11-14</td>
<td>2001 LITA NATIONAL FORUM - Milwaukee, WI. The Library and Information Technology Association’s annual discussions of how IT is transforming libraries and information technology. Contact LITA at 1 (800) 545-2433 x4270, fax 1 (312) 280-3257, or by email at <a href="mailto:lita@ala.org">lita@ala.org</a>. <a href="http://www.lita.org/forum01/index.htm">www.lita.org/forum01/index.htm</a></td>
<td></td>
</tr>
<tr>
<td>OCTOBER 14-17</td>
<td>EC’01 - Tampa, FL. ACM Conference on Electronic Commerce covers all areas of electronic commerce, focused on computer science issues, but interdisciplinary as well. Register online or email <a href="mailto:ec01.general@umich.edu">ec01.general@umich.edu</a>. General inquiries about ACM, call 1 (212) 869-7440. <a href="http://www.acm.org/sigs/sigeom/EC01/">www.acm.org/sigs/sigeom/EC01/</a></td>
<td></td>
</tr>
<tr>
<td>OCTOBER 15-18</td>
<td>FALL 2001 VOICE ON THE NET - Atlanta, GA. Pulver.com’s biannual Internet telephony fest. Learn about the latest developments in the world of convergence. For information, call 1 (631) 547-0800; email, <a href="mailto:von2001@pulver.com">von2001@pulver.com</a>, pulver.com/von</td>
<td></td>
</tr>
<tr>
<td>OCTOBER 18-20</td>
<td>SECOND KYOTO MEETING ON DIGITAL CITIES - Kyoto, Japan. Brings together computer science and social science people concerned with digital cities. Contact JST Digital City Research Center, 81 (75) 257-6214, fax 81 (75) 257-6216. General information, email <a href="mailto:meeting@digitalcity.jst.go.jp">meeting@digitalcity.jst.go.jp</a>. <a href="http://www.digitalcity.jst.go.jp/meeting/">www.digitalcity.jst.go.jp/meeting/</a></td>
<td></td>
</tr>
<tr>
<td>OCTOBER 19-21</td>
<td>NURTURING THE CYBERCOMMONS 1981-2021 - Ann Arbor, MI. Computer Professionals for Social Responsibility looks at the history and future of the global cybercommons, including Internet governance, “openness,” virtual communities, election technology, etc. Contact Nathaniel Borenstein, <a href="mailto:nsb@cpsr.org">nsb@cpsr.org</a>. For CPSR contact information, call 1 (650) 322-3778, fax 1 (650) 322-4748. <a href="http://www.cpsr.org/conferences/annmtg01/">www.cpsr.org/conferences/annmtg01/</a></td>
<td></td>
</tr>
</tbody>
</table>
Calendar of High-Tech Events

OCTOBER 19-21  POP!TECH 2001 - Camden, ME. This year’s title is “Online, Everywhere, All the Time: How it Will Change Our Lives.” Speakers include Carl Yankowski, Nadine Strossen, John Naisbitt, and Bob Metcalfe. For more information, call 1 (207) 230-2425, or email info@poptech.org. www.poptech.org/home.cfm

OCTOBER 23-27  WEBNET 2001- Orlando, FL. A world conference on the WWW and Internet, sponsored by The WebNet Journal and organized by AACE. For information, email conf@aace.org, or call AACE at 1 (757) 623-7588. www.aace.org/conf/webnet/

OCTOBER 31 - NOVEMBER 4  ICEC 2001- Vienna, Austria. The first International Center for Electronic Commerce conferences to be held in Europe. Special emphasis on EC firms and government bodies concerned with e-government. To register, contact Johanna Leithner, 43 (1) 515-14242, fax 43 (1) 5124-226, or email johanna.leithner@intropa-dmc.info.tuwien.ac.at/icec2001/

NOVEMBER 4-6  FM 1 NEW DEFINITIONS: VALUE COMMUNITY SPACE- Heerlen/Maastricht, The Netherlands. Focuses on the impact of “digitization” and the themes of value, community and space. Register online at www.infonomics.nl/newdefinitions/. For more information, contact Kamini Aisola at kaisola@kaiaconsult.com. www.infonomics.nl/newdefinitions/

NOVEMBER 5  SECURITY AND PRIVACY IN DIGITAL RIGHTS MANAGEMENT- Philadelphia, PA. One-day workshop as part of the Eighth ACM Conference on Computer and Communications Security. Contact ADM at 1 (212) 869-7440, fax 1 (212) 944-1318 or register at www.star-lab.com/sander/spdrm/.

NOVEMBER 5-9  NEXT GENERATION NETWORKS 2001- Boston, MA. The 15th annual conference organized by John M. Quillan and Business Communications Review. Covers the alphabet soup of networking technologies, including broadband access, wireless and optical networking. www.ngn2001.com

NOVEMBER 7-9  EDVENTURE'S HIGH-TECH FORUM- Berlin, Germany. Get connected! Our 12th year in Europe. Call Daphne Kis, 1 (212) 924-8800; fax, 1 (212) 924-0240; daphne@edventure.com; www.edventure.com

NOVEMBER 16-18  INTERNATIONAL CONFERENCE ON ONLINE LEARNING - Orlando, FL. The 7th Sloan-C International Conference on Online Learning: “Emerging Standards of Excellence in Asynchronous Learning Networks.” For information about conference arrangements, contact J. Patrick Wagner, 1 (407) 207-4920, fax 1 (407) 207-4930, or email jwagner@mail.ucf.edu.

Events Esther plans to attend.
Events Kevin plans to attend.

Lack of a symbol is no indication of lack of merit. The full, current calendar is available on our Website, www.edventure.com. Please contact Karen Schrier (karen@edventure.com) to let us know about other events we should include.
The conversation continues! Our free email newsletter offers commentary, industry analysis and pointers to interesting Websites on a regular basis. To sign up, please visit http://www.edventure.com/conversation/join.cfm or send email to conversation@edventure.com and you’ll automatically be added to the list.

Do we have your email address? Release 10 subscribers can now download each month’s issue electronically. If you have not already done so, please send your email address to natasha@edventure.com or fax it to 1(212) 924-0240 in order to enable online access.

---------

Release 1.0 Subscription Form

Complete this form and join the other industry executives who regularly rely on Release 1.0 to stay ahead of the headlines. Or if you wish, you can also subscribe online at www.release1.0.com.

Your annual Release 1.0 subscription costs $795 per year ($850 outside the US, Canada and Mexico), and includes both the print and electronic versions of 11 monthly issues; 25% off the cover price when you order from our online archives; a Release 10 binder; the bound transcript of this year’s PC Forum (a $300 value) and an invitation to next year’s PC Forum.

NAME

TITLE

COMPANY

ADDRESS

CITY STATE ZIP COUNTRY

TELEPHONE FAX

EMAIL* URL

*personal email address required for electronic access.

My colleagues should read Release 10, too!
Send me information about multiple copy subscriptions and electronic site licenses.

Check enclosed      Charge my (circle one): AMERICAN EXPRESS MASTER CARD VISA

CARD NUMBER ________________ EXPIRATION DATE ________________

NAME AND BILLING ADDRESS ___________________________________________

SIGNATURE

Please fax this form to Natasha Felshman at 1(212) 924-0240.

Payment must be included with this form. Your satisfaction is guaranteed or your money back.

If you wish to pay by check, please mail this form with payment to: EDventure Holdings, 104 Fifth Avenue, 20th Floor, New York, NY 10011 USA. If you have any questions, please call us at 1(212) 924-8800; email us@edventure.com; www.edventure.com.