



# **Web Services: A Floor Wax or Dessert Topping?**

**A Business Opportunity White Paper**

**Barbara Angius Saxby**

**March 7, 2002**

# Web Services: Floor Wax or Dessert Topping?

**Barbara Angius Saxby**

Lately you can't go anywhere without hearing about Web Services. Web Services have become the topic-of-the-moment in most trade rags, lunchtime discussions at software companies, and VC conferences. What is all of this hype about Web Services? What *IS* a Web Service? Why is it important and who should care? And most importantly, what are the opportunities for start-ups? These are a few of the questions Accelent has been investigating over the past several months. We have attended conferences, poured over white papers, talked to and worked with software vendors, and swapped opinions with VCs as we try to understand where it will all lead. To answer the question about what "it" is, I was reminded of a popular analogy from a Saturday Night Live skit, that promoted a product by stating "it's a floor wax AND a dessert topping" meaning it serves all of your needs. The hype around Web Services also proposes to solve all our computing needs – which is not the case. But, it does promise to be a pretty good floor wax.

The 2001 mantra was that the adoption of Web Services within the enterprise would result in more flexible application infrastructures, easier application integration, reduced development time and costs, and collaborative commerce with partner and suppliers. As we begin 2002, the projected opportunities for this technology, presumed to be infinite in 2001, seem to be hitting a more realistic plateau. A healthy skepticism has developed about the technology and its promise. Much of the debate around the topic of Web Services centers on its role in the enterprise infrastructure and in its application to e-commerce. Where does the real value lie? To answer this, you can't survey users because very few actual implementations of this new technology have been deployed. But large established market leaders and early stage start-ups alike are developing product strategies to capitalize on the promise it brings. Is this rational planning for the future, or a defocusing distraction caused by the latest bright and shiny new thing?

Our intent in this paper is to provide an overview of Web Services and our view on its impact on the software industry. We hope to provide a few answers to some basic questions on what Web Services are, including a technical overview of the current standards. We also discuss the platform wars between J2EE and .Net and the key issues for IT departments that will impact adoption in the coming years. Lastly, we identify where we see the most interesting opportunities for software vendors and summarize our position on its impact on the industry.

## **What a Web Service Is - and Isn't**

The software industry has been a buzz the past year speculating about the next monumental shift in architecture that promises to dramatically improve infrastructure design, application development, enterprise integration, and 'service' deployment over the web. Driven by the demand for true collaboration between business partners and faster deployment of web applications, this new software architecture is intended to super-charge business interactions and increase efficiencies.

The technical definition of a Web Service, as described by Microsoft, is programmable application logic accessible through standard Internet protocols. Web Services combine the best aspects of component-based development and the Web. Like components, Web Services represent black-box functionality that can be reused without worrying about how the service is implemented. Unlike current component technologies, Web Services are not accessed via object-model-specific protocols but through ubiquitous Web protocols and data formats, such as Hypertext Transfer Protocol (HTTP) and Extensible Markup Language (XML). A Web Service can be implemented on any platform in any programming language, as long as the application can create and consume the messages defined for the Web Service interface.

Web Services promise to provide unprecedented application interoperability and cross-leverage by allowing them to participate more broadly as integrated components of complete e-business solutions. By exposing components of applications as Web Services, and enabling 'consumers' to invoke these components, businesses can strengthen their ability to integrate enterprise applications and interact with current and potential customers and partners. In this new distributed, service-based environment, transactions in the form of XML message exchange allow for just-in-time integration and deployment of modular bits of application logic for performing specific business tasks. Next generation Web Services will be described, published, discovered, and invoked at run-time in a distributed network environment.

The term Web Service is used by various groups to describe widely differing concepts. This has complicated discussions and market conditioning.

Infrastructure software vendors, like Microsoft, IBM, Sun/iPlanet, Oracle, and others are the primary vendors who claim responsibility for engendering the Web Services movement. They tend to support the traditional W3C (World Wide Web Consortium) standard definitions and view Web Services as a programming protocol that can expose aggregations of objects and content over the Internet, thereby turning the Internet into a dynamic medium for programmable information exchange.

This is the usage of the term Web Service that we are adopting in this paper. In its technical definition it is not a service you can buy, rent, or write a contract around. The term "service" used here is in a software architectural sense – what "service" or function does one hunk of software code provide to another if called upon to help out. This concept is as old as software is itself, from do loops to callable subroutines to object brokering. But now, the mechanism for invoking a service is via HTTP, so therefore we call today's architecture Web Services. And since the nature of this architecture engenders new methods for collaboration and new opportunities for injecting value into a business ecosystem, it could go full circle; with bits of application logic, objects, and content being created by entities and sold and bartered to other entities across a Web network fabric. If this happens, then this new software architectural concept will really earn its name.

The term Web Service is also used by ASPs to define specific applications that are built for web usage, packaged as a product, and delivered in a variety of service models. ASPs deliver a closed 'black box' application while Web Services are inherently extensible. ASPs offer a dynamic business model for hosting and delivering applications and are not fundamentally about the technology that is used. Web Services, the way we define them in this discussion, may enable some new business models, but they're fundamentally a technology definition.

Content delivery vendors and e-commerce payment processing vendors have defined what they provide as Web Services too. Their definition centers on the value delivered to the customer. In their view, these Web Services are applications -- pieces of business functionality -- served up over the Web.

The businesses that call their product Web Services today may leverage the technology that we are calling Web Services to enhance their Web Service product offerings in the future. Clear as mud, right? That's why there is often some confusion when discussing Web Services. From this point on we will use the term Web Services to refer strictly to the software architectural model described at the beginning of this section.

## **Web Services - Low Hanging Fruit**

The prevailing wisdom is that the ultimate adoption of Web Services in the enterprise will largely focus on helping to solve the B2B problem, where a need exists for low overhead solutions and 'ubiquitous' environments in which businesses can find each other (the Web, http) and perform some basic interactions over the Internet. The problems experienced to date in achieving B2B collaboration between partners were largely due to a lack of trust (and good security technology) by companies in exposing their databases and applications to each other. Centralizing the relevant information that companies choose to share, without exposing the entire application, is the goal for many software companies who will leverage the web in their B2B applications.

But, as we've seen, you have to get your own internal house in order to best capitalize on the efficiencies of B2B. Therefore most activity now (vendor positioning and product roadmaps) centers on how Web Services can be used to facilitate faster enterprise application integration. Usage in this domain offers the most short-term benefits if it can contribute to easing the pain of integration. It's a natural extension of middleware type architecture problems the industry has been trying to solve: how to best share data between applications and underlying architectures and reuse components. In fact, those of us that remember the advent of CORBA (common object request broker) and object-oriented programming began by scratching our heads as we attempted to figure out how different the Web Services paradigm is from these other standards.

Here's the answer: The introduction of object-based technologies in the early to mid 90's promised component reuse for application development. But the fact is, this technology is complex, hard to use, and requires technically savvy developers and architects. Web Services are an extension of this component-based model but the difference is that Web Services are loosely coupled (not tightly integrated) and they are built on existing ubiquitous protocols like HTTP and XML. The goal for Web Services is to enable businesses to access parts of an application and integrate it with data from other applications, business process platforms, or infrastructure functionality and bind them together to deliver a composite application or applet that can be accessed from anywhere using the Internet.

## **Standards**

So much of the promise of Web Services depends on broad interoperability between applications. Therefore, the make/break lies in the evolution of technology standards and much of the work today is in this arena. A brief summary of the technology is presented here and a detailed discussion can be found in Appendix A.

In the early days of the Web, core technologies were used to provide an interface to distributed services (e.g., HTML forms calling CGI scripts). XML has accelerated this development, and has sparked the emergence of numerous XML-based environments that enable Web Services. These environments are starting to encompass the classical components of distributed application environments, such as standard protocol conventions, security mechanisms, processes to ensure reliable delivery and provide transaction functionality, and interface description languages, all of which are adapted to the special needs of the Web environment, and the requirements of XML. There are a few key standard specifications and technologies involved when building or consuming Web Services. The goal of these technologies is to define:

- A standard way to represent data
- A common, extensible, message format
- A common, extensible, service description language

- A way to discover services located on a particular Web site
- A way to discover service providers

The 'official' source for defining the standards used for Web Services is the World Wide Web Consortium (W3C). It develops interoperable technologies such as specifications, guidelines, software, and tools and serves as a forum for information, commerce, communication, and collective understanding.

The industry is attempting to take advantage of accepted W3C standards such as XML, HTTP, and DNS protocols to define and agree on a Web Service platform. The basic data representation and translation platform is XML plus HTTP. HTTP is a ubiquitous protocol, running practically everywhere on the Internet. XML provides a metalanguage in which you can write specialized languages to express complex interactions between clients and services or between components of a service. The XML message is converted to a middleware request and the result is converted back to XML. DNS is the standard protocol for mapping the service delivery to deliver the message to a unique address on a server.

XML is the industry choice for a standard way to represent data. Most Web Service-related specifications use XML for data representation and XML schemas to describe data types. XML is a great foundation; it enables flexible encoding of almost any kind of structured data in a way that doesn't mandate any particular language or operating environment. But, unless both the sender and receiver of the XML message speak the same language or set the same translation requirements, interoperability between applications is difficult. XML protocols are very effective and offer a simple model for inter-application communication, but unless the applications involved know how to talk to each other, their ultimate value is limited. At a minimum, applications must know:

- How to find each other
- What sorts of messages are expected by the other side
- What kinds of transport protocols are in use
- How to provide the necessary information

To create 'true' Web Services, this basic capability needs to be augmented with a few other technologies, while maintaining the ubiquity and simplicity of the Web. The 'true' Web Services platform can be thought of as XML plus HTTP plus SOAP plus WSDL plus UDDI.

SOAP means *Simple Object Access Protocol*. It is used to encode messages such as remote procedure calls (RPC) for request-response messages. WSDL means *Web Services Description Language* and is used to describe a Web Service interface. In essence, SOAP and WSDL facilitate distributed computing. SOAP provides a simple distributed computing mechanism that can be used over multiple transports. It also defines a way to perform RPCs using HTTP as the underlying communication protocol.

UDDI means *Universal Description, Discovery and Integration* service. It specifies a mechanism for Web Service providers to advertise the existence of their services and for Web Service consumers to locate services of interest. Using a UDDI interface, businesses can dynamically connect internal applications and services provided by external business partners.

## **Future Standards Development**

As this industry emerges, we can expect a lot of changes and additions to the standards. And like other waves we have seen in the software industry, we may end up with conflicting standards that end up doing more harm than good. The W3C is addressing some of these issues and has identified the following areas for future standards improvements:

- Reliable messaging
- Security
- Privacy of business data
- Transactions
- Interface definition languages
- Discovery of Web Service applications
- Web Service descriptions
- Message and protocol semantics
- Development environments for Web Services

The skepticism surrounding a truly universal development platform remains high, as it has been an illusive goal since the inception of the computing industry. Many industry leaders (Sun, Microsoft, Oracle and others) have branded their own flavors of Web Services while others are holding onto the older technology paradigm of object-oriented programming and applying it to the new model. The promise of distributed re-usable components remains the same; the implementation may vary. If all technologists could work towards an integrated solution (including CORBA, COM, JINI, JNDI, and other distributed object related technologies) that would benefit the vision of true Web Services, these standards would move along more quickly. If the standards efforts at W3C do not proceed quickly, they will be marginalized and become roadblocks instead of building blocks.

## **The Platform Wars: Unix/Windows; J2EE/.Net – Here We Go Again!**

Web Services has fueled the fire between two different camps that have begun waging 'platform wars' to determine who will be the dominate framework in the new generation of distributed computing. And who better to wage the war than Sun and Microsoft. It is important to understand the reasons for this debate as it will impact developers, IT departments, technology partnerships and vendors that will be building products to support these different frameworks.

Fundamentally, J2EE (Java 2 Platform Enterprise Edition) and .Net are similar frameworks. Both support the foundation of programming languages, component models, and virtual machines (run-times). The difference is platform independence. The advantage of J2EE is that it offers a single programming language capable of running on multiple platforms whereas .Net offers multiple programming languages (C#,J#, VisualBasic.Net) but runs on a single platform – Windows.

.Net seems to have an early lead in Web Services since it executes XML messages through HTTP and has incorporated SOAP into its framework. However, enterprise IT departments tend to be more Java-centric given their need for highly scalable and robust infrastructures. And, after several years in development, J2EE is finally a mature and reliable platform and is becoming a de facto standard within the enterprise. Both .Net and J2EE provide viable solutions, but companies should select a dominant framework to foster seamless integration and application integration, and accelerate application development as a long-term strategic initiative. The two most

important criteria when making a platform decision is (a) the openness of the standards the framework uses and (b) the number of platforms the framework can deploy.

This war is definitely being waged on the battlefields of the development community. A challenge for software vendors is getting new recruits from the ranks to join the Web Services movement and they are positioning to create new technology loyalists. Java developers have been working with J2EE for years so they have a lead in understanding the nuances of the technology and will unlikely switch to Visual Basic.Net. In the other camp, the move to Web Services is creating a learning curve for millions of Microsoft developers to learn C# and/or Visual Basic.Net.

The industry is debating who will win the war. Based on surveys of the developer community and IT organizations, it seems likely the world will continue to remain divided and that both frameworks will survive and linger. We expect that J2EE will remain the dominant player in large enterprise environments and .Net will find some success there but will predominately be used in Microsoft-centric application environments. This assessment is not based on the technology but rather the division in the Java and Microsoft camps today among the user community. The goal of both camps should be to enhance the ease of communication and interoperability and strive to make this work easier – but this is a lofty goal that has been the cry of both Microsoft and Java developers for many years. Most IT departments currently run multiple environments and we expect that they will continue to do so. We expect that ultimately, the framework decision will be shaped by the nature of an organization's current environment and resources, and not on technical merit alone. The good news: competing technologies keep companies innovating and provide users with a variety of products and services to choose from.

## **The Enterprise IT Perspective**

IT managers have not yet been tasked with managing applications and business services based on Web Services -- but the story might be different 18-24 months from now. IT management is about ensuring the reliable delivery of highly available, high-performance business services to employees, customers, partners, and suppliers. Therefore, there are a few basic considerations that will affect decisions made by IT management in terms of Web Services.

If the task of the IT department is to supply of applications based on Web Services, then it needs to understand how these services will perform in it's software environment. It needs to cost-effectively manage availability and performance. If applications are going to dynamically search and select the Web Services they need in real-time, IT managers must ensure repeatable performance for users. If applications are going to advertise Web Services for external use, they will need to understand and oversee the security implications of doing that. In general, Web Services are just another infrastructure device that needs management. Integrated fault management, performance monitoring, and capacity planning are needed to ensure consumer satisfaction with the associated applications. And since these applications are generally mission critical, this is not a step to be taken lightly.

The bottom line is, Web Services will probably make the IT manager's job more complicated in the short-to-intermediate run. The complexity is a function of the time it takes to:

- Investigate new technologies
- Separate vendor hype from reality
- Determine which products fit the overall IT strategy
- Investigate Web Services avoidance strategies
- Characterize and test new applications

- Identify and integrate new tools to monitor and manage Web Services environments
- Train personnel
- Coordinate deployment across applications

This work needs to be done to gain the benefits of Web Services. Concurrently, standards are still being developed, underlying mechanisms like XML are maturing, platform wars are raging, and hundreds of vendors are making thousands of announcements and claims clamoring for attention. Almost every established software vendor has made some announcement about its Web Services strategy. They usually promote future functionality and offer a glimpse of the product roadmap; what it really means is that the company's current offering will soon comply with current SOAP and UDDI standards. It will be important for IT to understand the goal of the project, the nature of the service that needs to be deployed, and the long-term goal of the organization when choosing solutions.

Ultimately, Web Services may make IT life easier by bridging the gap between business decision makers (and the applications they require) and the IT department. Some vendors promise that the creation of Web Services applets will become so easy that business analysts can actually create them with drag and drop graphical tools, taking some of the burden off of IT and allowing for more rapid deployment of these services as discrete applications. This kind of vision requires an IT infrastructure that is highly flexible and extensible, as well as a business culture that is confident that it can rely on software to empower decision-making. This has been a lofty goal for years, but the technology continues to become more complicated and the learning curve is high.

It is important to reiterate, very few real Web Services deployments have been implemented on a large scale to date – this industry is being built on technology vision and market hype today with some early pilots in process. We expect that IT departments will beta test the technology with small non-disruptive projects in order to evaluate its greater potential. This is a smart thing to do before any organization re-architects any major applications or departments. Most of the early work in testing Web Services has been to evaluate how it performs in the context of application development and integration. We expect early deployments to be in this arena given how hard it is and how incomplete today's current solutions are in solving the whole problem.

## **The Opportunities**

The vendor community with its intellectual capital, strategic visionaries, R&D resources, and marketing machines are busy working on their strategies, positioning, product road maps and alliance strategies. The VC community is also actively looking for the next great technology to create a disruption in the market to capitalize on and build sustainable businesses. The immediate impact on software vendors when these major trends arise is it drives every supplier to sit down and figure out (a) how to position itself vis-à-vis Web Services, (b) the scope of the research and the development investment required to bring its own Web Services-based offering to market, and (c) when to time the product launch and roll-out. The resulting false starts, failures and apparent successes will all be lessons learned for those that follow, helping to accelerate the market towards future innovations.

We expect that the implementation of Web Service-based solutions will occur at different rates for different vendors and different buyers - depending on the level of criticality of the business need they address and the importance of Web Services to addressing the need. We expect the revenue opportunities for vendors to map the natural evolution of technology adoption. Historically, the first people to make some money are those involved in creating the enabling technology itself. But ultimately, the big money will be made by people who leverage the

technology to solve problems that transcend the technology, either by improving applications and business processes that exist, or by seeing the opportunity to create something of value that did not exist before. Our crystal ball tells us that revenue associated with Web Services will flow to participating companies in roughly the following order:

- Phase I: Infrastructure
- Phase II: EAI Integration
- Phase III: B2B and SCM Applications
- Phase IV: Enterprise Applications

### **Phase 1: Infrastructure**

As usual, the first people who make money on a new technology will be those intimately involved in the development of the technology from the ground floor up. Phase I vendors will develop and sell technology tools and frameworks for building and managing Web Services applications. Developers and architects are using several first generation J2EE tools to build applications today. Current marketing campaigns promote tools and frameworks that will provide the easiest and fastest Web Service development and implementation.

The challenge for established companies building infrastructure products will be to innovate fast enough to gain mind share and revenue with easy to use tools and frameworks. Application server vendors are optimistic about the prospect of Web Services and are positioning their products as the deployment platforms for the next generation of service components. They are vying to be Web Services command central (in their marketing literature) to connect all the pieces - applications, integration, and business process management. Even though these vendors have market dominance and are well entrenched in the enterprise, there are those who doubt these big guys will get it right and be able to successfully deliver on this promise. They "don't know what they don't know" about the difficulties of building J2EE applications and it will take them too long to build solutions the users and developers need. In tough economic times, these companies manage their businesses to return earnings and positive results. Their focus is on delivering products that return short-term dollars as opposed to spending a lot of R&D money on longer-term strategic initiatives that new architectures require.

We see opportunities for smaller vendors that can bring new products to market that are flexible, scalable, and compliment the functionality of the application server while promoting the vision of service oriented architectures. However, start-ups in this space will be challenged with building a sustainable business developing discrete tools only. We do expect that innovative start-ups will quickly move to develop capabilities that will add several pieces of the needed functionality to deploy Web Service applications and therefore define their place in the software mix. But will they be able to do so profitably? Ultimately, the start-ups who build the tools that meet the right needs, at the right time, will probably be gobbled up by larger companies who will in turn integrate their unique technology into larger solutions.

There is a niche opportunity for start-ups to develop solutions to specifically manage Web Services and their components. Particularly as these services are deployed in distributed environments, they will need to be managed centrally by the IT department. To the degree that Web Services functionality cuts across traditional enterprise software vendor boundaries, limitations will be exposed in the management capabilities offered by each of those vendors. Overlay solutions will be needed. This creates a start-up opportunity because of the innovation required and the need for a new third party to arbitrate between the functionality provided by a number of application vendors. We expect that a new layer of internal brokering functionality will

be added to the software stack that will direct messages and content use from enterprise applications to third party Web Service management applications. These management applications will provide real time and long-term visibility into the interactions between Web Services components, and improve the manageability of systemic issues like business process flow management, performance, security and access control, policy management, failure analysis, diagnostics, contract management, format rendering, and version control. There are a few early stage companies that are moving into this space and are building products today.

One of the things that will separate the winners from the losers in this space is how well the tools vendors help make Web Services more secure by adding digital certificates to the mix. Protocols and mechanisms to strengthen the security, reliability and workflow capabilities of Web Services are vital. There is work in progress, but the approaches of today might not be robust enough for the enterprise of tomorrow, which may cause interoperability or integration problems down the road. The goal is to make security a fundamental part of each Web Service while reducing the complexity during development. Both start-ups and established security vendors have the opportunity to score big by improving the confidence level of the security in Web Services.

## **Phase II: Enterprise Application Integration**

Although categorized here as Phase II, we expect the development and adoption of this market will evolve closely behind, or alongside, tools development, as application integration continues to rank as one of the top five pain points for enterprise IT departments. The ability to build and deploy Web Services is fundamental, but reliably getting them from one location to many locations is essential to making Web Services useful. Phase II of Web Services adoption is short-term but also has sustainable, long-term market potential. We should see vendor offerings and integration projects using Web Services within the next six to twelve months.

Current EAI solutions link existing, monolithic applications into a common infrastructure, while Web Services are designed to allow for smaller, modular functionality that can be assembled and reassembled into dynamic processes. EAI solutions enable discrete, pre-specified connections, while Web Services enable open-ended, one-to-many connections. Lastly, EAI solutions are complex infrastructures that require a significant commitment of strategy and resources, while Web Services can be deployed with incremental cost and effort.

Web Services will play an important role in enabling applications using different infrastructures to work together seamlessly and they will be able to do it quickly and cost effectively. They will be able to access pieces of data to build composite applications that can be distributed and executed as needed anywhere in the enterprise. Given the enormous amount of corporate data and business logic that resides in legacy mainframe environments, enterprises will need scalable and efficient ways to expose these core assets as Web Services and share them when and where they need them. For EAI vendors and systems integrators, who are still in the early stages of this burgeoning industry, there is a significant opportunity to leverage this technology to solve escalating integration problems. Web Services will allow them to take more extensive advantage of that base capability and expand their target market potential by becoming a critical part of an integrated application system and overall business process.

We expect to see established vendors re-work their current offerings to provide more flexible platforms that can accommodate the use of Web Services to offer more efficient integration solutions to their customers. The nature of Web Services promotes a distributed service oriented approach to integration and current EAI solutions feature "hub and spoke" architectures that centralize all of the business intelligence and management functionality; there are inherent limits

on scalability and flexibility and there are many vocal skeptics in the industry about the long term viability of this approach. Therefore, we see a need for integration technologies that offer lightweight distributed integration frameworks and platforms that will propel this industry ahead even further. This innovation will most likely come from the incumbents, but there may be room for start-ups to emerge if their solutions contribute to ease the pain and speed the process of integration. Innovators will offer powerful new distributed platforms or extend the basic functionality of current brokers, buses and adapters using Web Services. We expect that the presently crowded world of EAI vendors will slim down to a few big winners, and the winners will be the ones who best embrace and leverage Web Services.

Furthermore, enterprises will need easy-to-use visual tools to guide business decisions when assembling Web Services to streamline end-to-end business processes. We expect this will breathe new life into the market for pure play business process management vendors. In addition, EAI vendors have been extending their roadmaps to include BPM as they all strive to improve multi-step processes to achieve both time and cost efficiencies.

### **Phase III: B2B Commerce and Supply Chain**

We expect Web Services usage to facilitate supply chain management and will be an important component of B2B applications and business process management solutions. We called these enterprise application categories out separately in this analysis because of the complexity of the business relationships and critical requirements for data sharing, transaction processing and order and inventory management. Early B2B trials of applications using Web Services have been primarily by companies functioning as service providers rather than service consumers. Concerns about the limitations of the technology in terms of security appear to be confining most of these early test implementations among known and trusted trading partners.

Even though the demand for innovation in B2B and SCM applications is paramount and many vendors are working on solutions, we categorize this as Phase III along the adoption continuum – 18 to 24 months out.

Before B2B collaboration can be expanded, significant work is needed in building the underlying infrastructure and integrating business processes. B2B collaboration and commerce also share the same requirements for better integration to the back-end systems EAI vendors are working to provide. Herein lies the opportunity. Large organizations and smaller technology providers that can deliver Web-Service based technology solutions to accelerate security for B2B transactions, improve integration, streamline business processes, enable commerce through centralized and interactive catalogs and directories, and also include the customer in the entire process, will find success.

### **Phase IV: Enterprise Application Vendors**

Established enterprise application vendors are actively investigating Web Services implementations of their applications but will cautiously watch how quickly demand evolves before they re-architect their entire product on Web Services standards. Phase IV adoptions presupposes all new applications will be introduced with underlying Web Service architectures that easily interact with other internal and external applications. We don't expect this to be the norm for another 24 months. Meanwhile, vendors will use the latest EAI technology and work with their SIs to implement applications using Web Services technology for early adopter customers. They may evolve road maps and add basic Web Services functionality to existing

products, create more APIs, or offer tool kits to make it easier for SIs to expose data to build and deploy applications as distributed services.

By the time Phase IV arrives, new standards are likely to be on the horizon and vendors will continue to feel the pressure to keep up, as technology is always a moving target. However, the large application vendors are so entrenched in Fortune 2000 companies that they must continue to upgrade their technologies and deliver innovative products along the way to meet customer demands and maintain or establish their competitive edge. We expect the large application vendors will continue to own the enterprise and will ultimately reap the rewards. In order to stay innovative, we expect accelerated M&A activity as vendors buy versus build Web Services solutions and industry consolidation continues to reshape the software landscape.

### **Ancillary Business Opportunities**

The evolution of Web Services will create additional opportunities for ancillary businesses and models that can leverage Web Services technology for profit. Potentially, the biggest winners will be SIs. These companies are not traditional early adopters, as they prefer to work with large, established vendors that provide them with the necessary technology, training and support – and credibility with their customers. However, as enterprise application integration, B2B commerce and supply chain management are the prime areas where Web Services technology will be utilized, and these are the “big ticket” projects for SIs, we expect them to reap the rewards from Web Services in the long run. We understand how slowed IT spending has hit the large integrators hard and they are less likely now to invest the time and effort in new solutions. But SIs need to evaluate the technology and align with vendors today to invest in tomorrow and they could end up with the brass ring. Mid-tier SIs that use Web Services for discrete projects will rack up some early wins, achieve market leadership (particularly in vertical markets) and make some serious money.

Another opportunity to leverage Web Services exists with the oft-maligned Application Service Providers (ASPs). Using Web Services, ASPs can improve the performance of their applications by having critical components of it operate at the customer site temporarily as the need arises. Yet they can maintain complete control over the application architecture, performance, and security. This could reduce the costs of application delivery as well as improve performance, which may help ASPs capture customers and make a profit on each one. Since the ASPs can be in total control of the application architecture and deployment, they can deploy Web Services principles and techniques immediately, and not have to worry about having some of the standards and “platform wars” settle down, like the EAI and B2B vendors must. We do not expect this opportunity to be as large or as likely as the ones we discussed in detail above, but it, like brokering services, should be addressed in this overview of Web Services.

Brokering services are subscription based offerings that provide management services for composite applets. The purpose of this new ASP model is to reduce the cost and complexity of managing interactions between partners and customers. These new service companies would offer a wide range of applications and platforms to provide more secure, scalable, and reliable communications across partner and supplier organizations and with customers. They offer companies a low risk entry into the Web Services arena while the ASP provides control, data transformation, visibility and management of the interactions and service delivery. We recognize this is perhaps the most futuristic view of Web Services adoption as infrastructure and integration technology must be in place and current UDDI standards need a lot more work. Market adoption of these new models is expected to be 36 months out and we expect several iterations before any real winners emerge.

## **Summary**

To summarize, we want to make it clear that in our opinion, Web Services is *NOT* a market. It is the next wave in software technology, based on advances in XML and other standards, that will enable new ways for enterprises to develop, integrate, deploy and manage applications. Web Services promise to make the simple act of connecting different pieces of software significantly easier.

There will be initial demand for development tools, application delivery platforms and management tools to facilitate the adoption of Web Services. These company's products are primarily valued for Web Service technology innovation within the IT community. This is the start-up opportunity, for the most part. We expect some new technology leaders will emerge and money will be made here, but these will be niche markets contributing to the overall success of the industry with a constrained maximum size. Established EAI, B2B commerce, supply chain management, other enterprise application vendors and SIs should make the majority of the big money from Web Services. We expect that innovation will be rampant in the early stages as the vendors who cleverly use the Web Services architectural model to better deliver existing application types or to conceive of and deliver new form of applications will be the big winners. In fact, those who don't embrace this model may rapidly lose market position to those who do.

We believe that Web Services technology is still moving through the early adopter phase and will shift to a more mainstream phase by mid 2004 and become widespread by 2005. Given the current economic climate, IT departments will be cautious in making investments and they should be looking at Web Services based on a clear business value. IT management will look for sound investments, technical interoperability and business process integrations. The need for interoperability with existing infrastructure and applications -- whether these are based on Java technologies, Windows applications, or homegrown legacy systems -- will continue to drive purchase decisions for information technology. IT management will also be looking for quick return on investment where expectations are now three to six months to show return.

Therefore, we expect that early adopters will continue to use Web Services for organic, discrete internal projects. The next phase will see more systematic deployment within the enterprise for application integration. We also expect continued efforts in B2B and supply chain management solutions to achieve ubiquitous collaboration between partners and suppliers. Lastly, as companies are looking to increase control of their business, the need exists for various business units to be connected and we believe that the granular nature of Web Services solutions will increase their ability to do so.

In terms of which software vendors will win – the jury is out as there is a lot of innovation expected during the next few years. However, we predict IT departments will continue to look towards their existing software vendors for technology solutions, but they will test promising technologies that fill an existing gap. Therefore, we believe as technology is such a dynamic industry, there will be room for a few point solutions and innovative products from start-ups to be deployed quickly and cost effectively, with minimal disruption on the current infrastructure, which will deliver the ultimate goal of easing the pain of software development and application integration.

## **About Accelent**

Accelent provides strategic management services to help software startups accelerate marketing strategies, planning, and execution. With a focus on enterprise infrastructure and applications, Accelent consultants offer strategic counsel, assist with projects, and fill interim management roles on a full-time or part-time basis. Bridging the gap between strategy and execution, Accelent develops successful strategies and delivers results through highly targeted marketing programs, even with limited budgets. Accelent is the answer for early stage startups that don't have the internal resources needed to position and launch the company or a new product. Accelent also helps later stage companies fill gaps in the management team, research and develop new products or programs, or adjust marketing strategies. Because Accelent consultants have a solid understanding of technology, learning curves are short and they rapidly assimilate the challenges at hand. They also bring valuable industry experience and knowledge to kick-start new marketing initiatives at any stage of a company's development. Accelent delivers value through innovative marketing strategies, actionable plans, and successful program execution - with proven results.

## **About the Author**

Barbara Angius Saxby founded Accelent to help software startups accelerate marketing strategies, planning, and execution. She specializes in positioning and launching enterprise infrastructure and application companies. Barbara is a senior marketing executive with over 20 years experience in strategic marketing management and has done extensive work internationally. Her experience includes both strategic project work and interim positions as vice president of marketing at the following companies: Reactivity (application security), Wakesoft (adaptive development platform), Taviz Technology (EAI adapters) Kirus (service supply chain management), Extricity/Peregrine (B2B relationship management), ECNet (SCM Services), Annuncio/BrightInfo (e-Commerce software), uRoam (remote access), NetScreen (network security appliances), and Resonate (traffic management). Prior to her consulting career, which began with Acuitive in 1998, Barbara was VP of Marketing for several software startups focused on application development and she spent nine years at Dataquest in a variety of analyst and marketing roles in the US and Europe. Barbara has degrees in Psychology and Sociology from San Jose University and completed post-graduate business courses at UC Berkeley including software venture financing.

## **References**

<http://www.xml.com/pub/a/2002/01/09/soap.html>

<http://www-106.ibm.com/developerworks/library/ws-uddi4j.html?dwzone=xml>

<http://www.xml.com/pub/a/2001/04/04/webservices/>

<http://www.w3.org/2001/03/WSWS-popa/>

<http://www.w3.org/2001/03/WSWS-popa/paper04>

<http://www.w3.org/2001/03/WSWS-popa/paper21>

<http://www.w3.org/>

[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/websvcs\\_platform.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/websvcs_platform.asp)

## Appendix A

The following is a more detailed discussion of critical industry standards discussed in the paper. It should be noted that while vendors try to present the emerging Web Services platform as coherent, it's really a series of in-development technologies. Often there are, and may remain, multiple approaches to the same problem.

### **SOAP (Simple Object Access Protocol)**

SOAP and WSDL are document formats. SOAP is used to encode messages such as remote procedure calls (RPC) for request-response messages. WSDL is used to describe a Web Service's interface. In essence, SOAP and WSDL facilitate distributed computing. SOAP provides a simple distributed computing mechanism that can be used over multiple transports. It also defines a way to perform RPC's using HTTP as the underlying communication protocol.

Traditionally, distributed computing standards have been based on binary protocols that require a particular infrastructure. SOAP, in contrast, is flexible in nature. Any application that can open a connection to the server, pass an XML document, and receive one in return can be a client. SOAP is agnostic with respect to programming language, platform, infrastructure, and vendor.

Submitted in 2000 to the W3C by IBM, Microsoft, UserLand, and DevelopMentor, the further development of SOAP is now in the care of the W3C's XML Protocols Working Group. This effectively means that SOAP is frozen and stable until such time as the W3C Working Group delivers a new specification.

### **WSDL (Web Services Description Language)**

It is necessary to have a standard way to document what messages the Web Service accepts and generates the Web Service agreement (or contract). The WSDL is an XML-based contract language jointly developed by Microsoft and IBM. WSDL provides a way for service providers to describe the basic format of Web Service requests over different protocols. WSDL is used to describe *what* a Web Service can do, *where* it resides, and *how* to invoke it. UDDI registries describe numerous aspects of Web Services, including the binding details of the service. WSDL fits into the subset of a UDDI service description. Having this standard mechanism makes it easier for developers to access tools to create and interpret contracts.

WSDL defines services as collections of network endpoints or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract descriptions of the data being exchanged, and port types, which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitute a reusable binding. A port is defined by associating a network address with a reusable binding; a collection of ports defines a service.

### **UDDI (Universal Description, Discovery and Integration Service)**

UDDI specifies a mechanism for Web Service providers to advertise the existence of their Web Services and for Web Service consumers to locate Web Services of interest. Using a UDDI interface, businesses can dynamically connect to services provided by external business partners. UDDI provides three basic functions known as publish, find, and bind:

- Publish: How the provider of a Web Service registers itself
- Find: How an application finds a particular Web Service
- Bind: How an application connects to and interacts with a Web Service

A UDDI registry contains three kinds of information, described in terms of telephone directories:

- White pages: Information such as the name, address, telephone number, and other contact information of a given business
- Yellow pages: Information that categorizes businesses
- Green pages: Technical information about the Web Services provided by a given business

A UDDI registry is similar to a common object request broker (CORBA) trader, or it can be thought of as a DNS service for business applications. A UDDI registry has two kinds of clients: businesses that want to publish a service (and its usage interfaces), and clients who want to obtain services of a certain kind and bind programmatically to them.

For more advanced applications, technologies such as XAML, XLANG, and XKMS, may be also be integrated. Although not universally adopted as mandatory, these other services are being developed rapidly so readers should understand what they do.

### **XLANG (XML-based language)**

XLANG is an XML-based language that supports complex transactions that may involve multiple Web Services. XLANG is the business process automation language utilized by Microsoft's BizTalk Server. Automation of business processes based on Web Services requires a notation for the specification of message about exchange behavior among participating Web Services. An XLANG service description extends a WSDL service description with an element describing the behavioral aspects of the service. XLANG is expected to serve as the basis for automated protocol engines that can track the state of process instances and help enforce protocol correctness in message flows. XLANG makes a notation for expressing the compensatory actions for any request that needs to be undone. The Web Services infrastructure can leverage XLANG specifications to perform complex undo operations.

### **XAML (Transaction Authority Markup Language)**

XAML is intended to enable a whole new class of dynamic, highly distributed computer interactions known as "business web transaction processing" or BWTP, as distinguished from OLTP, or classical online transaction processing. Business web transactions involve Web Services from multiple organizations on the web and must coordinate the low-level operations of commit, cancel, retry, and compensate (undo or reverse), in order to ensure business-level transaction integrity. The XAML initiative is so named because it is an application of XML that defines transactional interaction among Web Services, based on interfaces as defined by the widely adopted XA (Transaction Authority) standard and by JTA (Java Transaction API).

### **XKMS (XML Key Management Specification)**

XKMS is an effort by Microsoft and Verisign to integrate PKI and digital certificates (which are used for securing Internet transactions) with XML applications. The key idea is to delegate the signature processing to a *trust server* on the Web, so that thin or mobile clients don't have to carry around the knowledge to do this themselves. XKMS relies on the XML Signature

specification already being worked on by W3C and on anticipated work at W3C on an XML encryption specification.

### **WS-License**

WS-License describes a set of commonly used license types (credentials that are signed assertions) and describes how they can be placed within the WS-Security credentials tag. Specifically, the WS-License specification describes how to encode X.509 certificates and Kerberos tickets as well as how to include opaque encrypted keys. WS-License includes extensibility mechanisms that can be used to further describe the characteristics of the licenses that are included with a message.

### **WS-Referral**

WS-Referral is a protocol that enables the routing strategies used by SOAP nodes in a message path to be dynamically configured. SOAP itself provides a distributed processing model where SOAP messages can have content destined for specific processing nodes. WS-Routing adds to SOAP the capability of describing the actual message path. WS-Referral provides a mechanism to dynamically configure SOAP nodes in a message path to define how they should handle a SOAP message. It is a configuration protocol that enables SOAP nodes to delegate part or all of their processing responsibility to other SOAP nodes.

### **WS-Routing**

WS-Routing is a simple, stateless, SOAP-based protocol for routing SOAP messages in an asynchronous manner over a variety of transports like TCP, UDP, and HTTP. With WS-Routing, the entire message path for a SOAP message (as well as its return path) can be described directly within the SOAP envelope. It supports one-way messaging, two-way messaging such as request/response and peer-to-peer conversations, and long running dialogs.

### **WS-Security**

WS-Security provides a security language for Web services. WS-Security describes enhancements to SOAP messaging providing three capabilities: credential exchange, message integrity, and message confidentiality. These three mechanisms can be used independently or in combination to accommodate a wide variety of security models and encryption technologies. WS-Security provides a general-purpose mechanism for associating licenses with messages. No specific type of credential is required. Message integrity is provided by leveraging XML Signature and licenses to ensure that all messages are transmitted without modifications. Similarly, message confidentiality leverages XML Encryption and licenses to keep portions of a SOAP messages confidential.

These are a small subset of Web Services standards; for a more complete set of standard definitions, visit XML.com. XML.com features a rich mix of information and services for the XML community.

The standards glossary can be found at <http://www.xml.com/pub/a/2002/01/09/soap.html>.