The Past, Present and Future of Web Services

"Those who find success using Web services will be those who understand the technology fundamentally: its motivations, the reasons why some components are winning out over others, and the likely course of maturity." – Uche Ogbuji, Sept 2002

Author:

Uche Ogbuji is a consultant and co-founder of Fourthought Inc., a software vendor and consultancy specializing in XML solutions for enterprise knowledge management. Fourthought develops 4Suite, an open source platform for XML, RDF, and knowledge-management applications. Mr. Ogbuji is a computer engineer and writer born in Nigeria, living and working in Boulder, Colorado, USA. You can contact him at uche.ogbuji@fourthought.com.

The Past, Present and Future of Web Services, Part 1

Introduction

Web services are somewhere around the crest of their hype cycle and currently the darling of the prevalent media. This cresting is like that of other technologies in that it precedes full development and maturity. Web services, an undoubtedly important technology regardless of media interest, have a good deal of development ahead of them. Those who find success using Web services will be those who understand the technology fundamentally: its motivations, the reasons why some components are winning out over others, and the likely course of maturity.

For this reason, I start with the history of Web services. This is no mere nostalgic side-trip: the business and technical environment into which Web services was conceived, and the various players that have waxed and waned in prominence in their history to date are likely to have a strong effect on the future of Web services. You can already see this happening with developments such as the emerging role of Organization for the Advancement of Structured Information Standards (OASIS) as incubator of security, workflow and transaction standards for Web services. OASIS was once seen as the very opposition to mainstream Web services.

The stage set for Web services

Distributed application development has been an important field ever since typical computing moved from encapsulated jobs on centralized mainframe computers to peer-networked minicomputers and workstations. As this developed, the IT manager was still usually master of all he or she surveyed. The strategic concerns of distributed development were confined to whatever issues allowed the IT manager to accept the necessary inputs and deliver the necessary reports to management. Keeping a well-integrated data center was key, and supporting heterogeneous platforms and environments not so important. The areas where such things were important, say the electronic data interchange (EDI) shop were usually completely separate entities within the organization.

For some reason the later minicomputer era is often omitted from histories of distributed computing, even though it is the scion from which almost all the fundamentals of Web services sprang. This is probably because it overlapped, and was overshadowed by the PC revolution, which didn't really figure significantly in distributed computing until the commoditization of the Internet. The distribution of services across multiple workstations, which was encouraged in minicomputers as a way to scale and organize the data center, required a variety of approaches to communications, both synchronous and asynchronous, and low-level facilities for integration with arbitrary applications. As a result, the prevalent architectures and operating systems of the period, such as DEC VMS, Tandem and HP and Sun UNIX variants developed distributed messaging technologies that would astonish today's self-styled pioneers with their sophistication.

IBM came a bit late to all this with MQSeries. Seeing requests for applications to work across minicomputer platforms and its mainframe platforms, it purchased the popular ezBridge software which could already do a lot of this and grew it into MQ, which is the main technology from this era to flourish to this day. But messaging often required novel ways of thinking for programmers bred on procedural languages, who wanted to simply have familiar function invocation systems for remote procedures. At the very end of the 80s, the Distributed Computing Environment (DCE) emerged with an initiative to standardize the various competing remote procedure call technologies. This effort consciously omitted messaging technologies, and for a variety of reasons, mostly political, never attained widespread industry support. By this time, a new (though not necessarily better) generation of distributed computing was emerging.

At this time the IT manager was moving into the executive ranks: information systems had become fundamental parts of a firm's strategy and a prodigiously growing part of firms' budgets. The realities of organization meant that computing in most companies stopped being concentrated in a single datacenter, and instead exploded into a variety of departmental systems. Systems integration became one of the most important considerations in the choice of technology and instead of standardization on a platform, IT managers looked for standardization on a networking technology. IT shops adopted object systems as the standard for software development because of the promised benefits of maintainability and code reuse which promised to slim IT budgets and improve return on investments. As a result the emerging distributed technologies took on a strong object flavor.

Common Object Request Broker Architecture (CORBA), like DCE, also came out of an industry consortium effort to standardize on distributed procedure technology: this time the large Object Management Group (OMG) which focused on requests on remote objects for cross-platform distributed applications. Because of its attempt to make issues such as object state and lifecycle management transparent, CORBA proved less scalable than messaging technologies and DCE. The same problem plagued Microsoft's answers to CORBA: the Component Object Model (COM) and the DCOM remote object protocol. At the same time, e-mail and the Web were proving themselves the most successful distributed architectures ever, and designers sought distributed development technologies that provided the looser coupling of messaging technologies and the ubiquity of the Internet. Standards that were embraced by all major vendors were another wish-list item as they would reduce the risk associated with choosing such technologies.

During this period other more specialized distributed technologies were emerging. The popularity of Java led to its specialized RMI protocol, and the niche successes of MQSeries led to Microsoft and Java messaging flavors.

So, as the new millennium approached, the stage was set for a new generation of distributed computing, based on information-systems needs and experience with network technologies to date. The prevalent needs are:

- Suitability both for distributed operation within an application, and the use of generic services across applications. In otherwords: the ability to support both software developers and systems integrators.
- Suitability for exchanges within an organization and between organizations, requiring cross-platform support and a data-driven focus.
- Concordance with existing Internet infrastructure as much as possible.
- Ability to scale as the number of nodes, heterogeneity of nodes and the complexity of each node's needs increase.
- Solid internationalization.
- Tolerance of failure. Networks where nodes are very tightly coupled together often suffer catastrophic failure when one node goes down. This is a serious problem for heterogeneous networks.
- Strong support in general software development and business workflow management tools from a rich choice of vendors.
- Suitability for the most trivial request/response exchanges as well as handling the most sophisticated orchestration, transaction and security concerns where necessary.

Web services emerge

In its own high-end UNIX platforms, HP trod a remarkable path directly from the minicomputer era of distributed computing to modern Web services. In the early to mid 90s, HP Labs began to explore how to reduce the well-known technical and cost problems of distributed systems. The resulting design principles which led to HP's e-Speak, released in late 1999, addressed most of the needs outlined at the end of the last section, and e-Speak emerged as probably the first Web services technology, and certainly the first commercially marketed Web services technology. It used generic protocols such as HTTP and an XML data representation to treat all manner of networked systems as "e-services" into which one could rapidly plug data streams. Unfortunately, because its vision is far more coherent than the current state of Web services, HP has recently suppressed e-Speak in favor of a more mainstream Web services offering.

The same insight into how to use HTTP and XML technologies to meet the above needs came to the less corporate UserLand community, whose leader, Dave Winer, led the development of XML-RPC, a very simple (the specification is only a few pages) system for calling functions on a remote server. XML-RPC is still very popular, especially in the open-source community where peer pressure and the development of many open and license-free implementations has led to high levels of interoperability, and low cost. Perhaps inevitably, given its shoe-string roots and early arrival, XML-RPC has some glaring deficiencies in meeting the needs I outline. For one thing, its bewildering insistence on ASCII strings alone (despite its using the well-internationalized XML) means it is really not suitable except in English-only contexts. It also has a rather haphazard choice of data types. But most importantly, it is confined to simple request/response services with very uniform and highly structured message types.

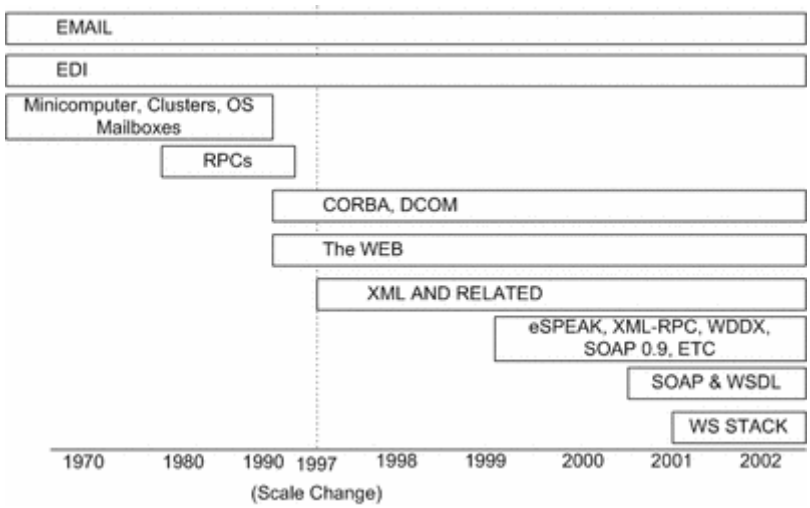
There were also more general messaging formats for XML emerging at about the same time as XML-RPC. Many of them were really more geared to the serialization of programming language constructs and database dumps than document-oriented

messaging, but they were still more flexible than the strict RPC approach. The most prominent of these is Web Distributed Data Exchange (WDDX), originally by Allaire, which is now an open specification with a significant community of its own.

Meanwhile, XML and Internet protocols were revolutionizing the thinking of a separate group as well: the EDI industry. Firstly EDI over SMTP (e-mail) and HTTP (Web) emerged as a way to dodge the expensive transaction fees of Value-Added Networks (VANs). Next, groups such as CEN/ISSS in Europe and the XML/EDI group in the U.S. (largely) started working on ways to encode EDI transactions, which are notorious for looking like line noise, as XML.

These trends led to early forms of Web services that were inspired not by the need for enterprise application integration (EAI) but rather for business-to-business transactions. These early XML/EDI systems were also designed to take advantage of the stable and well-understood EDI mechanisms for orchestration, security and other such matters that are still fuzzy in Web services space. The effort to turn this into a formal standard became e-business XML (ebXML), which, though mostly comprising the usual EDI stalwarts, was quickly joined by a vendor more usually associated with procedures and objects: Sun. ebXML, which started its self-imposed eighteen-month development period at the end of 1999, is a joint project of OASIS, an SGML/XML pioneer, and the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT), a key organization in the development of traditional EDI.

Also, back in 1998, a little specification for structured exchange of XML documents began brewing among a small group of kindred organizations (including the titan Microsoft): Simple Object Access Protocol was for various political reasons not released to the public in its original forms, and the SOAP we knew took the stage in late 1999.



HISTORY OF DISTRIBUTED COMPUTING

SOAP springs forth, and supporting languages proliferate

SOAP was born into controversy. From worries about the specialized type system it incorporated to the jockeying for position between the SOAP camp (publicly led by Microsoft) and the ebXML camp (publicly led by Sun), it was clear that the major players knew that Web services were the next big thing. SOAP also spread through the grass-roots Web services camp through advocacy that started in the same community as XML-RPC: Userland's. It was not as simple as XML-RPC, but it did address some of the earlier specification's shortcomings, and open-source developers were amazed to see all the major commercial vendors backing such an open specification (not all the surprise had been used up by the success of XML). Seeing the opportunity to wrench distributed development from the strict control of ponderous industry giants and consortia, a vibrant SOAP community sprang up.

SOAP is just a communications protocol. It became clear pretty early on that there would be significant value in a standardized form for Web services metadata, that is, the information that informs the processing of Web services. There were many initiatives for creating such a specialized language. Web Interface Definition Language (WIDL), by WebMethods, was one of the earliest ones, targeted as it was towards first-generation Web services systems such as XML-RPC and WDDX. WIDL is designed to mimic Interface Definition Languages that are the basis of CORBA and COM, but in XML form. Microsoft developed Service Description Language (SDL) and SOAP Contract Language (SCL) in 2000 for service descriptions of Web services end points. Microsoft also developed a system for syndicating information about Web services: Discovery of Web Services (DISCO), which would allow users to find SCL descriptions of services.

IBM created competing specifications: the Network Accessible Service Specification Language (NASSL), which paralleled SDL and SCL, and Advertisement and Discovery of Services (ADS), which paralleled DISCO (and was incorporated into the IBM alphaWorks toolkits for Web services. These, along with similar efforts from Ariba and other companies, coalesced into the Web Services Description Language (WSDL), originally by IBM, Microsoft and Ariba.

The same trio was also merging their various proprietary discovery systems into Universal Description, Discovery and Integration (UDDI), a system for directories (white pages, yellow pages and "green" pages) of Web services. UDDI was released as the product of a consortium of 36 companies, which soon grew to over a hundred. Its products were very elaborate and highly formalized, as distinct from the original SOAP and WSDL specifications, which were very simple and obviously written for straightforward understanding and implementation. This difference is a reflection of the realities of Web services: the low-level technical details such as the core communications and technical service descriptions were expected to grow from the bottom up, by incorporation into a variety of freely-available languages and toolkits, while discovery and business/legal description would have to grow from the top down, according to centralized authorities of information and certification of such information.

This has become a fundamental split in the "personality" of Web services. Firstly, there is grass roots Web services: the scrappy, "desperate Perl hacker" facet which has flourished within departments and separate from overall corporate strategies. Then

there is Corporate Web services: a cornerstone of e-commerce strategy, which is chartered in boardrooms. Many of Web services successes have been at the grass roots level, and UDDI has been put forth as an example of how Corporate Web services is yet to find its central value proposition (and how it has in some cases refused to learn the long and hard lessons of the EDI community before it).

Vendors take over the stage

The W3C Workshop on Web services (WSWS), in April of 2001, was a grand exercise in planning the future of Web services. Web services had so far been developed outside the W3C, either by the grass roots or by separate consortia such as UDDI.org. It seemed a natural move to have the W3C bring Web services under the aegis of other Web standards such as HTML and XML. The aim of the WSWS was to charter the form and goals of Web services activity in the W3C. The XML Protocol working group (XMLPWG) had already been formed in the W3C, although it had done little substantive work besides cataloguing the myriad of proposals and candidates for XML protocols. By the WSWS, it was already understood that the XMLPWG would be beefed up for focusing on the task of working SOAP into a full W3C recommendation.

True to the opportunities Web services was providing for reviving all manner of technology strategies, the WSWS plumbed a dizzying array of considerations to build upon the base protocol:

- IBM looked to emphasize network management issues such as quality of service (QoS) with its Web Services Endpoint Language (WSEL), as well as orchestration of Web services (formal description of the business processes underlying service exchanges) with Web Services Flow Language (WSFL).
- Hewlett-Packard offered the broader and more ambitious approach to Web services that characterized e-Speak, discussing the need for ontologies (formal, machine-readable descriptions) of the core concepts underlying any particular service, as well as the need to keep a variety of service environments in mind, especially the embedded space and small-device computing.
- Microsoft also looked to emphasize QoS, and along with IBM presented the vision of Web services that the two companies had been jointly pursuing.
- Jamcracker and BEA emphasized the importance of Web services in systems integration.
- Verisign emphasized the importance of security and the role of digital certificates. Novell emphasized the management of such authorization and certificates in directory services.
- In addition, many vendors were merely showing off their own proprietary frameworks for developing and deploying Web services as future models for any standards to emerge.

There was token representation of the grass roots arm of Web services (Dave Winer of UserLand, early developer of both XML-RPC and SOAP) as well as liaisons from the OASIS and ebXML. The latter had recently ended a major dispute in the Web services community by adopting a variation of SOAP over e-mail with attachments rather than a more elaborate specialized messaging protocol they had earlier developed. I myself was invited to present a paper on how the W3C's Resource

Description Framework (RDF) can and should be used for integrating the expression of Web services metadata with other metadata systems.

The struggle for the stack

The main theme of the WSWS was the Web services "stack", or architecture. Position papers suggested where various facilities, such as QoS, security, orchestration and transactions might properly fit into the stack. SOAP and WSDL were firmly entrenched as the core of Web services, so the battle to fill this stack with technologies favorable to the various major players took center stage in the latter half of 2002.

Security has been an area of especially concentrated work. It is well understood that before Web services become prevalent in communications outside corporate firewalls, strong security will have to be in place. There are many aspects of security, and one is just making sure the payload has not been tampered with. The W3C's XML Signature working group works on a system for digital signatures of XML documents, supported by specifications such as canonicalization (c14n). c14n converts XML documents to a standardized form so that the order of attributes and such things that do not affect the meaning of XML documents do not cause false positives for altered payloads. The XML Encryption working group works on encryption, including the XML Encryption Syntax and Processing specification, for encrypting data so that the result is a well-formed XML document. The XML Key Management working group works on simple Web services specifications for management of digital key and trust certification tokens. There are newer developments in 2002 such as Security Assertion Markup Language (SAML) and the WS-Security framework. I shall cover these in the next part of this article.

Business process and workflow are another part of the stack that have seen a lot of activity. ebXML is the early player here with its Business Process Specification Schema (BPSS) and its registry and repository standards for managing capability and agreement profiles for partners. It also provides long-lived transaction management support, security and QoS. Also early was the Transaction Authority Markup Language (XAML), by IBM, HP, Oracle, Sun and others, which has now melded into other Web services transactions efforts. IBM's WSFL and WSEL covered workflow and QoS respectively. Microsoft's entry, XLANG, covered workflow. WSFL and XLANG are both extensions to WSDL and, as expected, have been pitched in to a generalized business-process flow language WS-Coordination, which, along with WS-Transaction and BPELWS, I shall cover in the next part of this article.

And so to 2002

As we rounded into 2002, Web services were evolving at all levels. At the grass-roots level, where various developer interest groups had set up to sort out the notorious interoperability problems of early Web services. At the corporate level, the Web services stack was coming into place, and vendors were becoming impatient for the next wave of Web services specifications to give the final push to respectability. In part 2 of this article, I shall map out the present (year 2002) of Web services, and a look at what the future might bring.

The Past, Present and Future of Web Services, part 2

Part 2 - Introduction

2002 has been a year of consolidation for Web services. As I outlined in part 1, the interested parties and technologies have been converging, and the community is building on the established SOAP and WSDL core. IBM and Microsoft have been working together to fill out parts of the Web services stack; HP has moved from its original e-Speak product to a version based on SOAP, WSDL and UDDI; OASIS has also shrugged of its early indifference to SOAP.

However, one question still remains unanswered, will this convergence lead to robust, secure and interoperable solutions the industry so clearly needs? In this article I will take a look at how things are progressing on the standards front, where the issues and challenges are for the future and how they can and are being met. To appreciate what the future has in store Web services, it's important to appreciate the whole picture of Web services, such as its goals, its peers, and its user base.

The fiery forges of Web services

One clear point of diversity in Web services has emerged in 2002. That is, the forums in which they are developed. In February, IBM, Microsoft, Intel, BEA and other companies formed the Web Services Interoperability Organization (WS-I), a non-profit organization for promoting Web services standards. The idea behind WS-I was not to create new standards, but rather to assemble "profiles" of standards from the W3C, OASIS and other such bodies. The profiles are sets of related standards against which conformance tests and certifications can be established.

Despite such attempts by the big players such as IBM and Microsoft to avoid battles over turf, such battles immediately sprang up anyway. The development of Web services in the W3C has always been somewhat contentious. The various working groups involved are quite large, and general architectural stipulations of the W3C committees have slowed down development of even such key standards as SOAP and WSDL. This relative degree of control that organisations such as the W3C and OASIS have over Web services is seen as good news to some and bad news to others. The former feel it is better to develop Web services deliberately and with great care as to where they fit into Web infrastructure. Others are aghast to see the formerly prodigious progress of Web services slowed down in committee and lack of direct accountability. There has been a lot of grumbling among this latter camp that the W3C needs some impetus to get it to speed up development of Web services, and many see the WS-I as just such an impetus. Of course, this leads to complaints of the WS-I encroaching on W3C turf. Criticism has, however, been thrown at the WS-I in respect to its intellectual property rights for Web services protocols, and this is a big concern for some.

The debate between open and closed development is broad and far reaching, and only a few relevant points have been covered here. Ultimately, success can only be measured by the end result, that is, the protocol itself. Like all standards, the protocol will have to be developed through broad consensus, which will inevitably lead to results that do not please everyone.

OASIS, which stands for "Organization for the Advancement of Structured Information Standards" has become another outlet for parties impatient with the pace of Web services development at the W3C. OASIS offers openness, especially contrasting the WS-I, which is at the most shuttered end of the scale. Organizations and individuals can join at modest expense, and it is easy to set up a working group with just a few formal expressions of interest from members. Because of this, groups of interests have been set up which hope to standardize various aspects of Web services, such as security and transactions. Those groups are often working with the support of big companies and benefit from fairly complete protocols such as WS-Security donated from the likes of IBM and Microsoft. However, as with all committee based organisations, there is the lurking danger that progress will be slow. Significantly, UDDI has recently come under the umbrella of OASIS as well, and so the stakes are now even higher for OASIS to work efficiently.

Another continuing controversy has been the role of Sun in the WS-I. A consistent backer of the equally criticised and acclaimed ebXML standard, Sun led the original reluctance to SOAP, WSDL and UDDI, and also led the eventual reconciliation. Sun was left out of the WS-I and there has been on-going controversy over whether Sun should be accepted as a board member, including a bit of titillating press evidence that Microsoft was lobbying especially hard to block Sun's involvement. Sun, which is positioning its Open Net Environment (Sun ONE) as a key platform for Web services, is a significant player because of its leadership of the Java community.

A flashpoint that has developed quietly, but has also been cause of much agush is intellectual property rights. The W3C has traditionally required all contributions to its recommendations to be issued royalty-free. IBM and Microsoft, who have a large portfolio of patents relating to Web services technologies, pressured the W3C to change policy to allow reasonable and non-discriminatory (RAND) licensing and royalties. This was met with a popular outcry. The W3C patent policy is still under development, but in its current form, it leans heavily towards the royalty-free policy. In effect, the W3C Web services activity is a royalty-free group with no exceptions. Uncertainty as to the status of patents under the eventual W3C policy was one of the issues that led to the creation of the WS-I, which accepts RAND licensing of components of its standards.

Web services and its goals

Leaving aside the fiery forges of standards development, clearly Web services have gained a lot of mind share, but are they meeting the goals set by most technology movements? Are Web services meeting the particular needs that brought them about?

For a while they have had a bad rap on interoperability. SOAP implementations in particular were famous for their differing interpretations of the specification. This is one area where a lot of hard work has been done, especially at the grassroots level through the soapbuilders mailing list and other such groups. Recently, a high-profile group of developers set out to address the questionable reputation of Web services for interoperability with a panel discussion and demo of interoperability at conferences, starting with XML Web services One. This group started with a WSDL specification of a scenario for an e-commerce exchange involving actors such as consumers, warehouses and credit institutions. The various participants, including representatives

of WS-I and OASIS, IBM, Microsoft, each implemented different parts of the specified services using different tools, and presented a demo on the conference exhibition floor with end-points at each sponsor's booth communicating with the others. This initiative is similar to a touring interoperability demo by various companies involved in ebXML. The intent is to demonstrate that Web services tools do work well together, and for the most part, this was the general conclusion. However, not all was well, in particular there was a contradiction between respect for document-oriented messaging and reliance on toolkits. Do Web services teams need experts with detailed knowledge of the rich messaging involved in Web services? The panel was in disagreement.

Filling out the stack

For Web services to be useful, particularly in business processes, the stack must be filled. This is a matter that continues in development. Security and business process are key issues, as is integration into software development tools. Here is an overview of the current status of each.

Security:-

OASIS has taken under its umbrella many of the matters pertaining to Web services security. Major players have been vocal in complaints about the slowness of the W3C to begin work on security, and Web services security also involves high levels of intellectual property claims, raising the stakes. The OASIS Web Services Security Committee (WSS) consolidates work by IBM, Microsoft and others and focuses on digital signatures, encryption and the propagation of security tokens. The XML-Based Security Services Committee (SSTC) furthers SAML in standardizing the exchange of authentication and authorization information (which has some overlap with WSS tokens). The Rights Language Committee (RLTC) covers digital rights and permissions.

Business process management:-

Current developments in Web services business process and transactions include the release of Business Transactions Protocol (BTP) 1.0 by the OASIS Business Transactions Committee. BTP is a general XML-based protocol for B2B transaction management. IBM, Microsoft and BEA have merged their respective work in this area into WS-Coordination, WS-Transaction and Business Process Execution Language for Web Services (BPEL4WS). The first covers how business processes are coordinated within and across Web services. The second covers various forms of transaction management for Web services. The third covers the flow and procedure of general business process (and so has a large overlap with BTP). There is some indication these standards will be handed over to OASIS at some point. One of the important determinants of success in business process standards will be how general they are. IBM's and Microsoft's earlier offerings were strongly tied to their own software architectures such as Websphere, MQ Series, .NET and BizTalk Framework. Their combined standards are more general.

Component models:-

Web services have been incorporated into toolkits from the very beginning. Microsoft offers .NET and IBM the Web Services Toolkit; many older development tool companies and projects such as BEA, Iona and Apache now provide Web services features. This pervasiveness is the greatest success of Web services, but standards for guiding the integration into software development are still under development. At the high end of ambition in this space is the OASIS Web Services Component Model (WSCM) committee, which intends to develop a universal component model for Web services. This stretches from distributed network components in the tradition of CORBA and COM to visual components in the tradition of JavaBeans and Delphi. Some input in the later aspect of components comes from IBM in the form of Web Service Experience Language (WSXL), which aims at a way to specify user interactivity of Web services end-points. Epicentric's Web Service User Interface (WSUI) targets a similar space.

Web services and its peers

Understanding the challenges Web services face from the "fiery forges" of standards development, and additions to the protocol stack such that they can be genuinely useful in business processes, is only a part of the picture. You see, Web services are not the only revolution moving through information technology. Other developments from object repositories to Intelligent Web technology are finding greater interaction with Web services, and may shape important frontiers in their future direction.

Web services and Open source software

Open source software (OSS) has grown to extraordinary strength recently, and the OSS community has had strong involvement with Web services from the beginning. OSS developers have led the grassroots arm of Web services, which has scored some of the most significant practical successes. However, this interaction has not been entirely harmonious. The OSS community has been wary of Web services as a vehicle for subverting the next-generation Web for commercial interests. The OSS community also led the outcry against the spectre of royalties and licenses that could be imposed for patented Web technologies. These two issues are still being worked out. Many commercial vendors have released "community" versions of their Web services toolkits. Others have rather presented their frameworks as open to OSS implementations, such as .NET which spawned the OSS Mono implementation.

Web services and the Intelligent web

"Intelligent Web" is an umbrella term for technologies that improve the Web. An ambitious effort along these lines is the Semantic Web (semweb), led by Tim Berners-Lee, inventor of the Web and director of the W3C. The Semantic Web intends to give the Web some of the advantages of an integrated knowledge-management system. One of the disputes in the W3C has been whether it puts too much effort into semweb activity, which is seen by some as rather academic, instead of Web services activity. Semweb development has had some influence on Web services technology, including the assignment of Uniform Resource Identifiers (URIs) to service resources in versions of SOAP and WSDL created in the W3C activity. At the grassroots there have been efforts to bridge semweb and Web services, I myself being a frequent contributor in articles such as Managing structured Web service metadata and Using RDF with SOAP (RDF is a key semweb technology).

Web services and the OMG's MDA

The OMG's Model Driven Architecture (MDA) is an approach towards developing large-scale software that starts from the basic concepts that make up the problem space, expressed in a formalized but implementation-neutral manner. The OMG is a very influential organization, with strong support in a variety of industry verticals, and there will probably emerge an effort to develop WSDL and SOAP implementation bindings for MDA models. Web services are already finding bindings in component repositories from vendors such as Borland (Delphi), Microsoft (VisualStudio), Sun (NetBeans) and IBM (WebSphere).

Web services and Grid Computing

Grid computing is the idea of tying together numerous and diverse computing devices together to create a sort of virtual supercomputer where processing cycles and other resources can be shared for attacking overall tasks. Some companies, such as Sun and IBM, have made aggressive moves in combining grid computing and Web services. The Globus Project, a broad, open development group for grid computing, offers mechanisms for offering grids through Web services. The next version of the Globus Project, version 3.0, is to be built on the "Open Grid Services Architecture" or OGSA for short. OGSA is based on Web services and builds upon SOAP and WSDL.

However, one issue with Web services is the communications overhead, and partly because of this concern, Web services have been positioned as a gateway to grid systems rather than as the core communications layer for such systems.

Web services and other technologies

Another important convergence will be between Web services and alternate computing technologies. Many recent technologies, from Jini and JXTA through to Wireless Applications Protocol (WAP), offer standard systems for integration and communications in devices from high-technology printers to PDAs. Web services could bring such systems the benefits of technology interoperability and convergence. When communications go through a conventional computer, standard off-the-shelf Web services tools can be used, reducing cost and increasing options. Again, the inefficiency of Web services is an issue. The overhead of including an XML parser may be too much for some embedded applications, and the XML (and maybe HTTP) going over the wire or spectrum can be expensive given that bandwidth is often at a premium in such applications.

Web services against its user base

In the previous section I considered some exciting future applications for Web services. However for Web services to be widely adopted one must first understand who the adopters actually are, and their motivations. At this early stage, adopters of Web services have come in two major groups.

Transparent approach to Web services

One group merely looks at Web service technology as a useful feature of each entity's development toolkit of choice, designed to interact with other tools. The early success of this transparent approach to Web services has a large dependency on the political clout and fortune of the maker of the toolkits in question. Because of the gaps in the Web services stack and the rapid pace of change in the space, vendors often have to often provide avant-garde features and facilities to complement the basic protocols.

Such features are then nominated to standards efforts, causing competition and overlap. The winners of these standards sweepstakes thus directly affect users of affected toolkits. In this environment, homogeneous shops which are loyal to a particular vendor are usually at lower risk. This also favors internal applications of Web services within the corporate firewall.

Opaque approach to Web services

The second group of Web services adopters follow an opaque approach, focusing on the well accepted and standard core protocols rather than their implementation in any particular toolkit. In this group, service development starts with general WSDL descriptions, for example, rather than with a proprietary interface description system that binds to WSDL. Gaps in the Web services stack are filled with general framework plug-ins rather than assumptions of a particular vendor's feature set. This allows this group to weather the volatility and politics of Web services better, but it takes a very highly trained and specialized team to make this possible. Developers in this group need to be experts in technologies as diverse as XML and XML Schema and Internet protocol details. The need for integration of Web services between multiple organizations sometimes forces an opaque approach.

Web services will have to develop in a manner complementary to both groups

Both groups of users are likely to persist into the future. Certain organizations and users will always have such specialized needs and resourceful development skills that they will continue to use Web services in opaque manner. Transparent usage will also spread as maturity, interoperability and availability of toolkits increase. This means that Web services will have to develop in a manner complementary to both groups. In particular, successful Web services technologies will be those that can be plugged into the widest variety of tools and standards without strain. Rather than a classic "stack" of layered technologies, Web services will develop into a network of multiple connected technologies. In such a network, marketplaces will form among natural clusters of shared need and interest. This is a relatively novel characteristic in the development of standards frameworks, but one that has derived from XML itself. Rather than growing grassroots or proprietary beginnings into highly structured and cross-dependent standards, XML and Web services are emerging as loose networks of relatively independent pieces. This is also behind their significant affinity for connecting to foreign technologies. Unfortunately, this also makes the choices harder for decision-makers in considering profiles of Web services technology.

Reading the Web services tea leaves

This article has covered a lot of ground, and it is now the time to draw some conclusions and make some predictions for the future. What has been demonstrated is that Web services standards need to progress through quickly and efficiently, they need to achieve their goals of interoperability, and more so, fulfil their potential by filling out the protocol stack. They need to be open and royalty free for wide adoption, and work towards complementing other new technologies. They also need to develop in a fashion that fundamentally suits both types of user base.

Web services in their present form will not be the media darling forever. How does one know what is still volatile or mere hype, and what will extend well into the

future? Certainly Web services are here to stay. The history of distributed computing shows that this area is especially sensitive to critical mass, or what is often called "network effect". The more broadly and deeply a technology is deployed, the more likely it is to dominate an era of computing. Web services have found critical mass greater than any other distributed computing technology before them.

But there are still choices to be made within the world of Web services, especially in these formative times. Surprisingly, one cannot always weigh such choices merely by the interests behind them. UDDI is an instructive example. It emerged to grand fanfare, and its stakeholders succeeded in anointing it into the accepted core of Web services technologies. "SOAP, WSDL and UDDI" was how Web services were described for much of the past two years. Despite this, UDDI has not taken off in practice, and many have begun to question whether it ever will. If it does, it will not likely be with its original ambition.

The factors that will determine the success of Web services technology start with their affinity across a variety of scales. Can the technology be used in the simplest grassroots project as well as the most complex business interchange? SOAP and WSDL pass this test well; UDDI, at least initially, is suited mostly to the higher end of the corporate Web services. One way to test this is to see whether the development of the technology involves both grassroots and corporate players. Does the technology build directly on the strengths of complementary technologies beyond Web services? These could include the Web, XML or general application development trends. Is the technology fundamentally extensible? How hard is it to add bindings, human- or machine-readable notations that allow customization and specialization? Do toolkits readily support such extensibility?

The dynamics of commercial competition will certainly always be a major force in the shaping of Web services, but big commercial interests have tried forcing their way before in distributed computing, without the sorts of success and opportunities they find in the current era. This is because of the diversity that has braced Web services. The health of this diversity will continue to be the dominant indicator of the future of Web services technologies.