Bloor Research N.A.

Bloor Research North America
34 Dartmouth Street
Yarmouth, ME. 04096

Phone (207) 228 4944
BloorResearchNA@CS.com

Web Services Gotchas

How Enterprises Can Build Secure, Reliable, Performance-optimized Web Services Solutions While Waiting for the Standards to Mature

Table of Contents

Figures, Graphics, and Drawings

Preface

Web services are an evolving series of standards that enable programs on various computers to communicate with other programs on similar or disparate computers transparently over the Internet.

The whole idea behind Web services is not new – in many respects it's just another distributed computing approach in a long line of distributed computing architectures. But, in many other important respects, Web services are distinctly different than predecessor distributed computing and program-to-program communications such as electronic data interchange (EDI), Common Object Request Broker (CORBA), Advanced Program-to-Program Communications (APPC), et al. ***The key difference is that Web services architecture does not rely on "hard-wired" connections between applications (this has a profound impact on how applications can be designed and deployed – and will result in the formation of completely new business models as Information Systems designers learn to exploit this new architecture).*** This concept of hard-wired versus the Web services loosely coupled relationship between applications will be explored in depth in this report. Just be aware – this loosely coupled approach to building and deploying new "service-oriented" applications is big news – it could fundamentally change the way the information systems are structured and thereby allow for changes to the way businesses are organized (and how they flow work). **As a result, Web services have the potential to change the very business models that today's businesses are constructed upon.**

If you believe that Web services will, over the next several years, dominate the program-to-program communications scene – then you may be tempted to build some experimental Web services applications today. And if you choose to become an "early adopter", you will soon learn that **Web services have some maturing to do.** Today, Web services architecture is good at sending and receiving "messages" to/from other programs – but **they lack "mission-critical" features in the areas of security, reliability, manageability, and scalability, as well as need improvement in the areas of transaction-handling, tuning and performance**. And without improvements in these areas, Web services will continue to be used as it is generally used today – largely as a messaging architecture for performing very simple message/transaction-oriented applications in non-mission critical program environments.

This report has been designed to provide you with an understanding of:

- what Web services are;

- where the shortcoming are in Web services architecture;
- what is being done by standards committees to address these shortcomings;
- how vendors are "*stepping-in*" to help fill-in-the-gaps until Web services standards mature; as well as
- provide you with basic selection criteria that can be used to help sort out the differences between various vendor's Web services offerings.

It has been designed to help you and your organization determine "when" is the right time to implement Web services within your own enterprise.

We hope that you continue to read this important report and take advantage of months of Bloor research and analysis designed to provide you with an appreciation of the state-of-the-technology of today's Web services architecture.

Who Are Bloor Research?

Bloor Research has long been recognized as one of Europe's top technology and business research and analysis firm. Founded by Robin Bloor in 1989, it consists of almost a dozen market research analysts who are recognized as thought leaders in their chosen technology practices.

In April, 2002, Bloor Research expanded its operations to include North America. (now known as Bloor Research NA). By so doing, Bloor positioned itself to provide end-users and vendors with a good understanding of how new technologies are being used on "both sides of the pond". To rapidly build its North American competency, Bloor recruited and hired Joe Clabby (former Group Vice President of Platforms and Services at the Aberdeen Group) to head its North American operations as president. Joe brings over twenty-five years of computer industry experience and a successful background building research and analysis organizations to the Bloor team.

We see our job as a provider of analytical services. We specifically seek to help Information Systems (IS) buyers understand evolving technology trends as well as understand how to best exploit those technologies. Our research is aimed at enabling our clients to quickly understand a given technology — its strengths as well as its limitations. By so doing, we believe that we can save our readers a lot of research, analysis, and learning time.

Our job is to save you and your organization a lot of time and effort in researching and analyzing Web services and its respective strengths and weaknesses.

At present our research is focused on next generation computing infrastructure (the systems, applications, and management architectures that make distributed computing possible). More specifically, in North America we focus on grid computing, Web services, and system/application/content management.

How Is This Report Organized?

This report is organized into three sections:

1) **A Quick Primer** – Part I of this report defines Web services and what they're being designed to do. It also identifies various Web services architectural "shortcomings". And it provides insight into the Worldwide Web Consortium (WC3) and other standards bodies that make recommendations related to Web services architecture. It also examines the activities of various vendors, consortia, and open source activities that are working to address Web services architectural shortcomings.

2) **The Current State-of-the-Technology** – Part II is designed to more closely examine previously identified Web services shortcomings. This section describes the areas in which the standards committees have focused. It also describes where Web services architecture "needs improvement" and how vendors are providing products that supplement Web services architectural shortcomings until standards recommendations are made and can be implemented.

   This section considers the status of each topic (such as security, reliability, manageability, etc.) – and is organized as follows:

   - *The Goal* – (for instance, "transactional security' or "routing/-messaging");
   - *The Issue(s)* – (issues that must be addressed to make the identified Web service enterprise-ready);
   - *Standards Activity* – (a description of where Web services standards organizations have focused their efforts in order to remedy architectural shortcomings);
   - *Remedies* – (a description of how certain vendors, open source organizations, standards organizations, and consortia are addressing identified architectural shortcomings).

3) **Vendor Listing** – Part III has been designed to help you understand what products various vendors offer to fill-in-the-shortcomings in current Web services architecture. It looks at the various product offerings provided by well known companies such as Microsoft, Sun, HP, IBM, Tibco, Iona, webMethods, and others – and includes short opinions of these vendors and their products. But it also looks at the products offered by some of the smaller vendors — vendors like Polarlake, Zebrazone, CommerceQuest, The Mind Electric, and so on — vendors that have integration frameworks or point product solutions that help address Web services shortcomings.

Note that the vendor listing at this juncture is not an exhaustive list of vendors that supply Web services products and services. But it is a start – and Bloor Research intends to expand this list as well as provide in depth research and analysis on many of these vendors during the course of 2002.

What You Will Learn

Our approach is to describe the standards activity and focus in a particular area of Web services (for instance "security"). We then describe what needs to be done to make Web services security enterprise-class for use in robust production computing environments. Finally, we wrap each chapter with a discussion of what vendors are doing to supplement/augment Web services until the security standards have been officially defined and implemented

When all is said and done (when you've completed reading this report) you will have learned:

- What Web services are;
- How they will be used;
- Where they need to be augmented in order to be deployed in mission-critical computing environments;
- Who some of the vendors are that offer solutions that augment Web services architecture;
- When to implement Web services within your own organization.

Comments/Feedback

We at Bloor Research North America are definitely interested in your feedback regarding this report. We readily accept both positive and negative feedback — and will do our best to incorporate your opinion in future revisions of this report. Please feel free to contact Joe Clabby, President of Bloor Research North America with your comments and/or criticism at:

BloorNA@AOL.com

.

Executive Backgrounder/Summary

There are dozens-upon-dozens of definitions of Web services. Some people see Web services as a group of communications protocols and an application interface sanctioned by the W3C (Worldwide Web Consortium) – as well as a related registry service being developed as an independent initiative. Others view Web services as a distributed systems architecture that enables object programs to provide "services" for one-and-other (computational, transactional, messaging, etc.). Still others define Web services more broadly as any architecture for passing XML (extensible markup language) documents and data between disparate systems and operating environments.

The bottom line is that *Web services are an evolving distributed computing architecture that uses its own program-to-program interfaces, communications protocols, and a registry service to enable similar and disparate applications to communicate and perform "services" for one-and-other.* The way that *these Web services applications* couple and work together (when linked to business process management software and process flow) *have the potential to radically affect how information systems are designed and operate.* Accordingly, Web services applications will most likely cause massive changes in existing business models across multiple industries over time (in an evolutionary, not revolutionary fashion)**.**

To understand Web services, therefore, it is necessary to understand distributed computing architecture, business process management, the applications-as-a-service concept, and W3C standards and directions. This report explores all of these topics.

Web services Definition — The Standards

In the description of Web services above, mention is made of certain programmatic interfaces, communications protocols, and a registry service. Today, Web services make use of SOAP protocols and WSDL interfaces — as well as in some cases UDDI registries to perform distributed computing functions. These interfaces/registry are defined as follows:

- Universal Description, Discovery, and Integration (UDDI) is a "registry" that allows applications to be listed and located. At present, UDDI is not a standard, but instead a joint initiative of businesses and vendors;

- WSDL (Web Services Description Language) is a template or interface that allows applications to describe to other applications the rules for interfacing and interacting;
- SOAP (Simple Object Access Protocol) provides the basic program-to-program remote procedure calls that enable applications to invoke each other and commence program-to-program communications.

What's So Special About Web services?
Joe Clabby, (one of the authors of this report as well as president of Bloor Research NA), describes in detail the benefits of Web services architecture to business in-depth in his new book — "Web Services Explained" (Prentice Hall, July 2002). In this book he claims:

Web services:

- *enable application developers to stop worrying about systems infrastructure and instead focus on writing cooperative applications* – Web services combined with XML enable applications to be written in any of the most popular programming languages and deployed on any of the most popular systems platforms. This makes much of the complexity in writing program-to-program applications (applications that can work cooperatively with each other – usually across systems environments) transparent and thus greatly simplifies the job of applications programmers. (At long last, programmers can focus on writing useful applications and focus less on making those applications work with underlying systems infrastructure).

- *help cut down on application development time and expense* (because developers can make use of existing applications modules rather than having to write new service applications from scratch every time they write a new program);

- *enable application writers to dynamically grow application portfolios* (in other words, it enables application developers to more-quickly-than-ever-before assemble *compound application solutions* using internally created "modules" as well as externally created modules to build compound applications quickly – thus enabling enterprises to bring new products and features to market more expeditiously).

- *open new marketing opportunities for enterprises that already have applications*. Existing applications can be sold as Web services modules (providing a potential new source of revenue to the enterprise);

- *enable enterprises to respond more quickly to changing market conditions* (new buyer trends, for instance); or to respond more quickly to competitive pressures; and,

- *help consumers and business users get access to highly personalized applications* that can run on multiple different devices (PCs, handhelds, smart phones, etc.).

And these are only but a few of the benefits offered by the Web services application development and deployment model! There are many other advantages related to how Web services can be used to create new go-to-market approaches, new efficiencies, reduce costs, repackage existing products or intellectual material, etcetera. Web Services Explained focuses on nine of these advantages (see Figure 1). It also provides practical examples of how Web services are being used in the real world (by businesses and enterprises as well as software vendors); as well as how they could theoretically be exploited as the technology matures.

Figure 1 — Nine Ways to Advantageously Use Web services

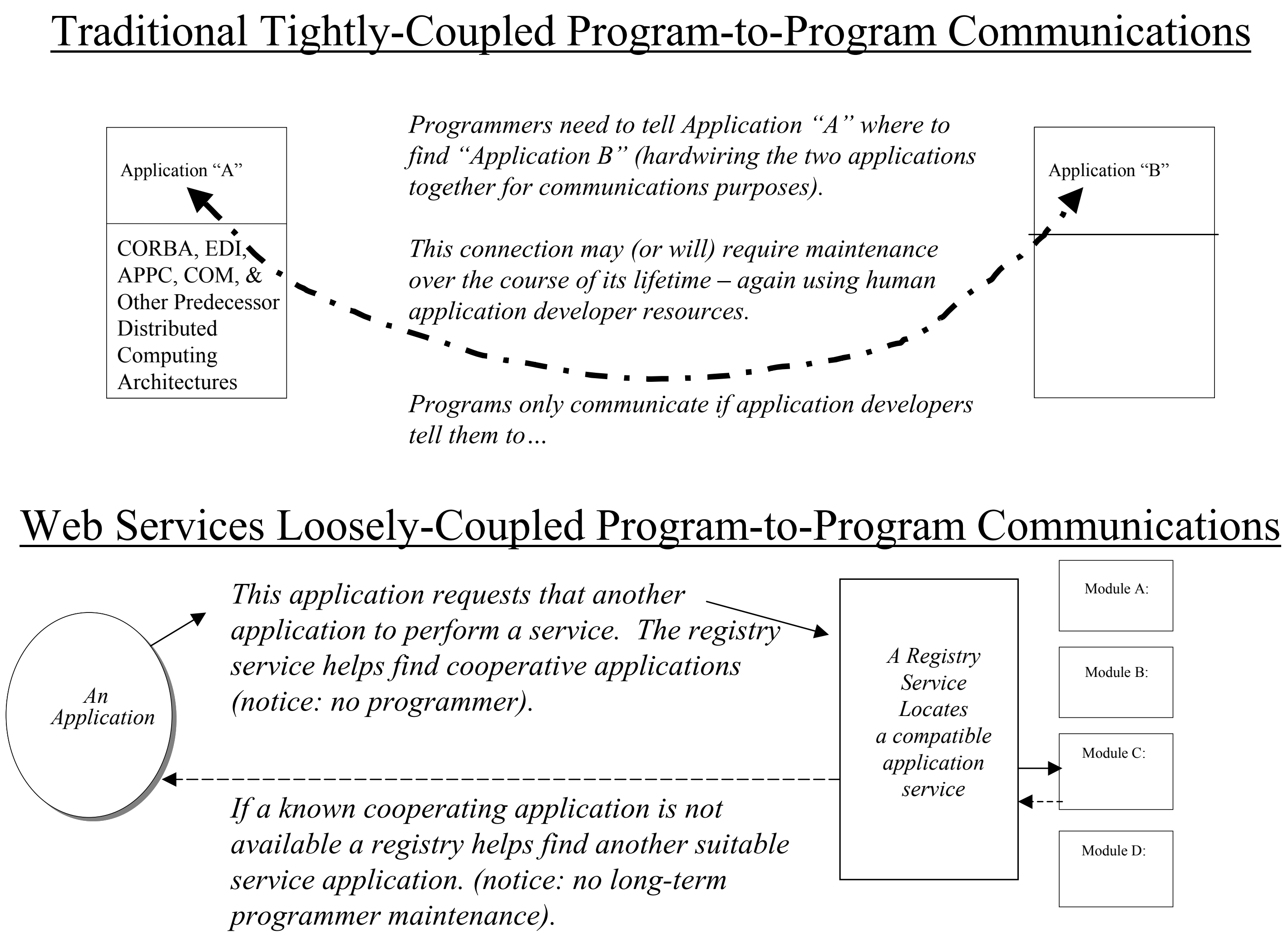
Nine Reasons to Embrace Web Services Architecture

1) Rapidly open new markets;
2) Create new organizational efficiencies;
3) Reduce application development costs;
4) Create/overcome competitive pressure;
5) Create new sources of revenue using existing intellectual capital;
6) Repackage existing products to better reach/serve existing markets;
7) Solve systems inoperability problems;
8) Dynamically grow an existing applications portfolio;
9) Increase/improve organizational and individual productivity.

Source: Web Services Explained (Prentice Hall, July, 2002)

What's Special About Web Services as a Distributed Computing Technology?
Probably the most important concept to understand about Web services is the concept of tightly coupled versus loosely coupled applications (see Figure 2).

Figure 2: Tightly- Vs. Loosely Coupled Application Behavior


Traditional Tightly-Coupled Program-to-Program Communications
Application "A"
CORBA, EDI, APPC, COM, & Other Predecessor Distributed Computing Architectures
Programmers need to tell Application "A" where to find "Application B" (hardwiring the two applications together for communications purposes).
This connection may (or will) require maintenance over the course of its lifetime – again using human application developer resources.
Programs only communicate if application developers tell them to…
Application "B"
Web Services Loosely-Coupled Program-to-Program Communications
An Application
This application requests that another application to perform a service. The registry service helps find cooperative applications (notice: no programmer).
If a known cooperating application is not available a registry helps find another suitable service application. (notice: no long-term programmer maintenance).
A Registry Service Locates a compatible application service
Module A:
Module B:
Module C:
Module D:

Source: Bloor Research NA – May, 2002

In short, the design of traditional distributed computing applications relies on having a programmer "tell" applications where to find each other in order to share information or perform cooperative tasks. This "hard-wiring" of applications has its benefits (for instance, it is easier to perform security checking if each application knows the other). And it is easy to fix a broken application because you always know where the cooperative application module resides. Using a tightly coupled application development approach provides certain safeguards from a quality-of-service, security, privacy, data integrity, and complex transaction processing perspective as compared to Web services architecture. Tightly coupled applications know the location of the applications with which they work (and this implies a certain "security" guarantee). Tightly coupled applications know how their partner applications behave (and this implies an ability to ensure a reliable conversation between applications as well as the ability to ensure performance characteristics to a certain degree). Tightly coupled applications are inherently more easily managed (because both endpoints are known). And for all of these reasons (security, reliability, service guarantees), the tightly coupled approach has dominated enterprise computing (both inside the enterprise as well as externally with B2B business partners).

But this "hard-wiring" has disadvantages too. Applications need to be told by a programmer how to find each other, how to communicate with each other, and they also require long-term management/repair. It is often difficult and cumbersome to build tightly coupled applications because programmers need to spend a lot of time defining the connections and relationships between cooperating applications.

On the other hand, the loosely coupled approach to designing and maintaining applications has other benefits. Building loosely coupled applications is simpler because developers do not need to spend a lot of time defining where cooperative applications can be found and defining the rules that allow them to communicate. Maintenance of loosely coupled applications may also be easier — for instance, should one side of a tightly coupled application break – both sides of application become broken (a scenario that can be easily overcome using Web services because a replacement application can be sought dynamically and run automatically). Loose coupling of applications also provides a level of flexibility and interoperability that cannot be matched using traditional approaches to building highly-integrated, cross-platform, program-to-program communications environments.

As IS buyers weigh the trade-offs between these loose coupling and tight coupling, they may find themselves coming to the following conclusions:

- Tight coupling is comparatively cumbersome (but is inherently reliable, secure, and tunable);

- Loose coupling provides benefits such as dynamic lookup and heterogeneous, cross-platform interoperability (but may require an organization find and integrate supplemental software for security, reliability, manageability, and other mission-critical purposes).

Which approach should your organization choose? The answer is not binary – both approaches are valid and appropriate — but some applications are better suited for tight coupling (for instance, heavy batch-oriented applications and financial transactions) while others loan themselves nicely to loose coupling (message-oriented applications). Each approach has it merits. But, from Bloor North America's perspective, we expect that once Web services architecture matures a bit and gets better at handling complex transactions in a secure fashion, this architecture will directly rival traditional distributed computing architectures such as Electronic Data Interchange and Common Object Request Broker and soon come to dominate our approach to application building and deployment.

What Else Is Special About Web Services?

Aside from the tightly coupled/loosely coupled discussion (above) – it is also important to note that Web services are special because they can perform:

- Dynamic look-up: Applications can automatically, *dynamically*, find other applications and establish a working relationship with such applications; and,
- Cross-platform, program-to-program communications. Yes, other architectures are capable of performing program-to-program communications – but Web services are endorsed by over five hundred vendors and end-user organizations. Other distributed computing architectures of the past have tried and failed to achieve heterogeneous systems interoperability (due to proprietary extensions that hindered program-to-program communications, or due to complexity). But Web services (using XML as the common "format/syntax" language and using WSDL and SOAP to facilitate heterogeneous communications are simpler to implement (in no small part due to the advantage of using a loosely coupled architecture).

More on Dynamic Function – By making use of UDDI registry services, loosely coupled "Web services" applications can automatically find cooperative partner applications (no human direction is required). And loosely coupled applications can negotiate how to communicate with each other (no human direction is required). And loosely coupled applications can automatically find other cooperative applications if their primary partner application should "disappear" (again, no human intervention required). (See Figure 2 for a comparison of the "traditional" approach to building distributed tightly coupled applications versus the evolving "Web services" approach using loosely coupled applications).

As a result of minimizing the amount of human intervention needed to enable applications to work cooperatively together, this dynamic aspect of the loosely coupled approach to building and maintaining applications holds great promise for streamlining and simplifying the development of cooperative distributed applications in the future. And, even more exciting: these loosely coupled applications will affect how businesses build and use applications – resulting in the creation of *dynamic new business models* based upon the rapid and fluid assembly of application services!

More on Heterogeneous Environments – *This point is very important because most of today's Web services implementations capitalize on the heterogeneity aspect of Web services architecture.* Web services enable programs written in differing program languages on differing systems platforms and operating environments to interoperate. For instance a C-language-based program could communicate with a COBOL-based program by using XML to present data, using WSDL to help determine how the two programs could interact, and using SOAP as the middleware program-to-program glue between the differing applications. In short, heterogeneous program-to-program communications support enables interoperability between disparate systems and applications. Many attempts in the past have been made to achieve cross-platform program-to-program communications – but Web services appears to be making strong headway toward making cross-platform interoperability a reality.

Web Services "Gotchas"

For Web services to reach their fully potential, certain "shortcomings" or "gotchas" need to be addressed. Bloor Research NA has identified seven "gotchas" within Web services architecture, including:

- Security/privacy
- Messaging/routing;
- Quality-of-services/reliability;
- Transaction-handling;
- Managability;
- Performance/tuning; and,
- Interoperability

But, despite the fact that we have identified and isolated shortcomings in these areas, we still believe that this architecture can be used in mission-critical computing environments if enhanced with third-party software or hardware/software appliances available from various vendors or through open source software suppliers. In short, we see that all of these shortcomings are in the process of being addressed by the following types of organizations:

- Standards committees (the Worldwide Web Consortium – W3C; National Institute of Standards and Technology – NIST; the Internet Engineering Task Force – IETF; and others);

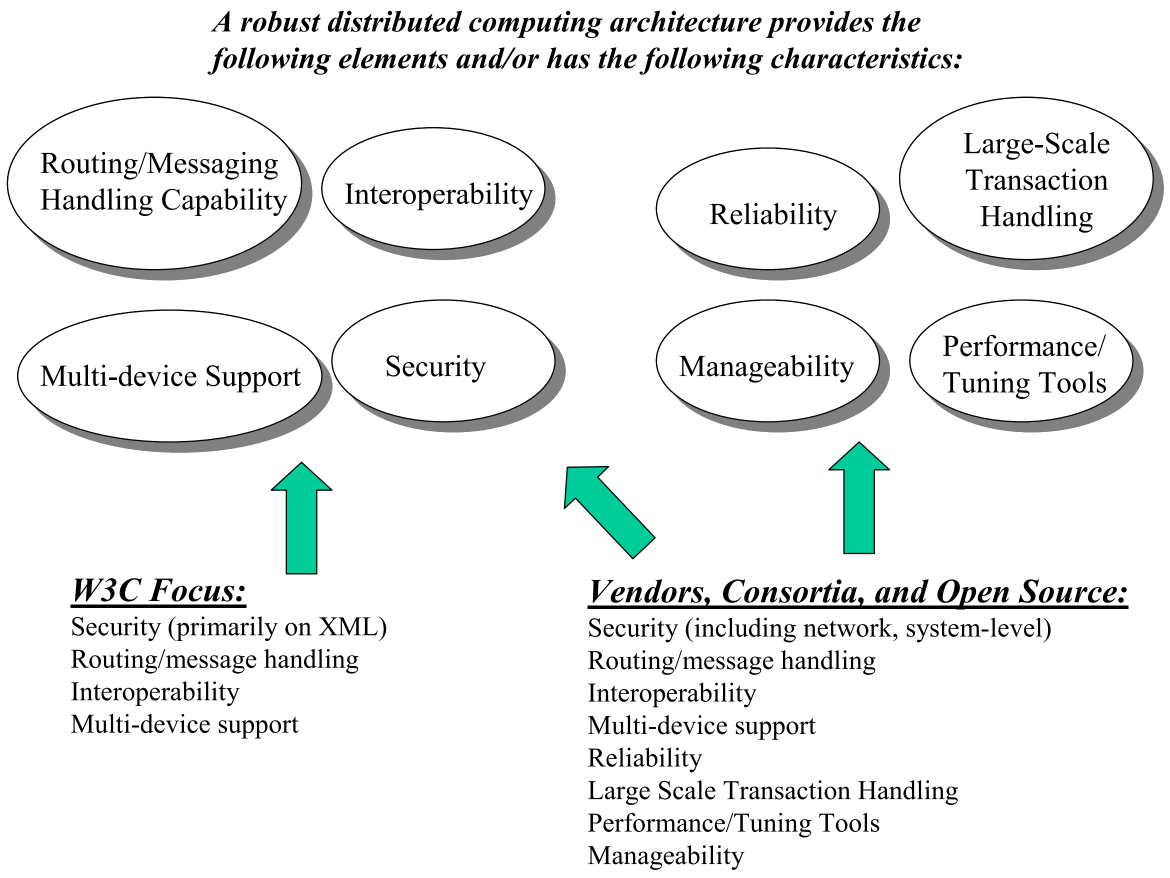
- Vendors (IBM, Microsoft, Sun, Hewlett-Packard, et al);

- Open source application developers; and

- Consortia (including the Web services Interoperability Organization – WSI; Organization for the Advancement of Structured Information Standards – OASIS; Open Buying on the Internet – OBI; Rosettanet, and others).

In graphics form, we believe that "the big picture" of how Web services shortcomings are being addressed looks like this (see Figure 3):

Figure 3 – Organizations Working to Resolve Web Services Shortcomings


A robust distributed computing architecture provides the following elements and/or has the following characteristics:
Routing/Messaging Handling Capability
Interoperability
Reliability
Large-Scale Transaction Handling
Multi-device Support
Security
Manageability
Performance/ Tuning Tools
W3C Focus:
Security (primarily on XML)
Routing/message handling
Interoperability
Multi-device support
Vendors, Consortia, and Open Source:
Security (including network, system-level)
Routing/message handling
Interoperability
Multi-device support
Reliability
Large Scale Transaction Handling
Performance/Tuning Tools
Manageability

Source: Bloor Research North America, May, 2002

Our research indicates that by mixing and matching Web services standards-based product with "enhancers" (other open source or vendor produced products that fill-in-the-gaps in Web services architecture), reliable, secure Web services environments can be built. And, by using messaging appliances, performance tuning software, and other hardware and software products – performance-related issues can be overcome (or at least mitigated). As a result, Bloor Research NA believes the following:

Even though Bloor Research NA has identified several Web services shortcomings that must be addressed by formal standards activity over time, it is our opinion that these shortcomings are not showstoppers – and we advise that enterprises not wait to experiment with and build Web services solutions. Between the advances being made within the W3C standards committees, and the advances and extensions being built by the open source and vendor communities – very functional (even mission-critical) Web services environments can be built and deployed today.

We do observe that, at this juncture, Web services architecture is best deployed in-house (on a corporate intranet) or on a protected virtual private network (with known secure business partners) because HTTP-based Web services can leave open holes for external hacking/invasion to take place. We also believe that at this juncture, Web services routing/message handling (and transaction handling) is not well suited for handling heavy transactional environments unless third party products (like appliances that off-load message handling and perform security tasks, or transaction monitors) are used to augment simple Web services routing and messaging.

Still, by taking these issues into consideration (and by using workarounds to overcome these issues) there appears to be no need to wait several years until the standards mature before experimenting with Web services architecture. <u>Web services architecture can be used successfully today in enterprise-class, production computing environments (provided it is properly augmented with the right mix of vendor-supplied hardware and software extensions to make it secure and reliable and capable of handling a lot of message traffic).</u>

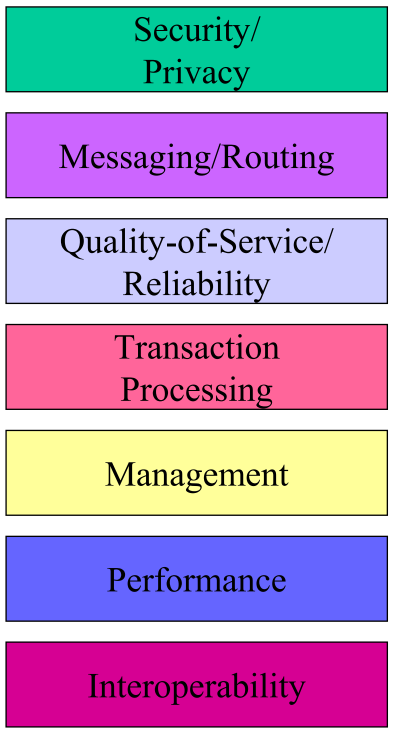
The Remainder of This Report

The remainder of this report describes what Web services are and how they can be used – with an eye toward helping you determine "when" is the appropriate time to deploy Web services within your own enterprise. It also examines each Web services "shortcoming" identified above – and details how standards committees as well as vendors/consortia/open source organizations are addressing those shortcomings. Finally, this report provides a listing of various vendor offerings in the Web services space – including commentary and critique on those vendor's strategies and product offerings.

Key Findings

Bloor Research NA has identified seven areas within Web services architecture that need to be made more robust in order to make Web services ready for deployment in mission-critical, production-mode, transaction-oriented enterprise computing environments (see Figure 4). We refer to these areas as present day "shortcomings" or "gotchas".

Figure 4 – Web Services Shortcomings/Gotchas According to Bloor Research NA



Source: Bloor Research NA North America, July, 2002

Progress toward creating standards for each of these gotchas varies greatly by topic area. For instance, great progress is being made by the W3C in specifying standards for digital signatures and XML content encryption – while there appears to be little progress being made from a standards perspective in manageability. From Bloor Research NA's perspective, each of these topical areas warrants a closer look:

Security Overview
Setting up enterprise-strength security relies on a combination of strong security architecture, products, and best practice security practices. From a standards perspective, the W3C recognizes that these elements are crucial to building secure computing environments – and addresses each element in the following ways:

- *Architecture* – the security framework in Web services architecture focuses on six elements: accessibility, authentication, authorization, confidentiality, integrity, and non-repudiation.

- *Products* – the W3C creates a number of open source "products" (such as "Jigsaw", its proof-of-concept advanced Web services server environment). But the W3C does not focus on building "point product" solutions (specialized products to address specific needs such as security management, application management, and the like). Instead, the W3C makes standards recommendations and vendors build product solutions based on those recommendations.

  A close look at evolving vendor solutions indicates that vendors are generally taking the following approach to securing Web services architecture: 1) they implement the standard recommendation, and 2) they add extensions (other products that can work over or with Web services architecture that help augment (make enterprise capable) Web services architecture. These extensions include mail/messaging server appliances with security, business process management, application development environments, portal services, and more. For instance, IBM's Tivoli products work with Web services and add additional security and manageability features that augment Web services architecture and make it more suitable for mission-critical computing environments. Other vendors such as McAfee and Forum Systems combine Web services standards for authentication, authorization, and non-repudiation with message-handling hardware and software – thus creating a security appliance that also handles message processing.

- *Practices* – Although the W3C is not in the process of forming standards for best practices in security, the site does contain a lot of useful product information related to considerations involved in building secure Web services environments. Bloor strongly recommends that readers who are interested in putting in place security policies and procedures for their Web services environments take the time to examine the following two URLs: 1) http://www.w3.org/Security/Faq/wwwsf1.html#GEN-Q7; and, 2) http://www.w3.org/Security/Faq/

The Use of HTTP

One weakness in Web services security lies in its use of the HTTP protocol. This protocol is very simple and straightforward (making it highly attractive as a means for applications to communicate with other applications over the Internet). But it can also be used to tunnel through enterprise security firewalls – thus creating a major break in enterprise virus and hacking defenses. Granted, much private Internet traffic is sent in an encrypted form using Secure Sockets Layer (SSL) for protection – and granted that many enterprises have security products and policies in place to help curtail HTTP security breaches – but Bloor Research NA is still concerned that using unprotected HTTP-based program-to-program communications represents a potential way to create security problems.

To help remedy this situation, Web services (specifically SOAP) can make use of other protocol stacks (UDP and TCP/IP) to allow Web services applications to enter an enterprise computing environment in a more secure manner (because Web services applications using these protocols would enter through a firewall) – thus helping to reduce exposure to hacking as well as viral attacks. But even this approach does not completely solve the need to better secure Web services transactions and messages.

The WS-Security Initiative

Perhaps a better way to address Web services security issues is to follow the security roadmap articulated by IBM, Microsoft, and Verisign. These companies have proposed a number of specifications that help strengthen Web services security (and have put forward a roadmap that can help enterprises address some of Web services security issues now – as opposed to waiting for the W3C security standards to gel). This roadmap – WS-Security – is covered later in this report in more detail.

Securing the Line

It is important to note that many enterprises have already gone to great lengths to protect communications in-flight over the Internet by using Secure Sockets Layer (SSL) encrypted protection at the communications/networking level. But fewer enterprises have taken steps to protect information at the data/content level. The W3C focuses a lot of its attention on protecting content – which Bloor Research NA considers to be a good use of W3C brainpower. Our reason: line protocols such as SSL and other security products and extensions are already available and are already used by vendors and enterprises to secure data-in-transit using encryption techniques over the Internet. Protecting content at the line level, and enriching it with features such as non-repudiation, authentication, authorization, confidentiality, etc., represents the next logical step in building an enterprise-class, secure architecture. And protecting content is very much where the W3C has focused its efforts as of late.

Finally, Bloor Research NA also notes that line security (SSL) and content security (authentication, digital signatures, authorization, etc.) are technology-driven security solutions. Also key to building secure environments are putting in place the proper policies and procedures to ensure that systems environments remain secure. Its one thing to have a lock (technology) on one's house for security – and quite another to fail to have a policy in place that prevents occupants from opening the door without ascertaining who's been knocking. Policies and procedures are just as important as technology to assuring that environments remain secure.

Reliability/Quality-of-Service Overview

In most of today's distributed computing environments IS managers can exercise a great deal of control over the systems and applications within their information infrastructure. But this scenario changes when key service applications are supplied by outside service providers who are beyond the control of the enterprise. For Web services to be used in mission-critical computing environments ways need to be found to ensure that service applications are always available when needed (or that alternative sources for application services can be found should problems arise with original service applications).

Our search for W3C activities related to reliability and quality-of-service found its way to the W3C "activity" (the way a W3C standard gets started) for Electronic Commerce. This activity group initially carried the responsibility for examining how reliability could be achieved (particularly for conducting business transactions over the Internet using Web services architecture). After an initial flurry of activity, this group apparently realized how difficult it would be to try and set e-Commerce standards in place because the requirements for e-Commerce differ greatly by industry. Hence the e-Commerce activity group chose to focus on defining core infrastructure technologies for achieving reliable transactions (and the Electronic Commerce activity group was subsequently closed). Reliability standards now manifest themselves under the auspices of other standards setting initiatives, for instance in the XML Signature, XML Encryption, XML Protocol, Semantic Web, Privacy, and Micropayment initiatives.

What this all means is that the W3C helps build the basic architecture to conduct secure, multi-phase, reliable transactions – but it relies on industry-oriented groups to help set in place the schema that enables e-Commerce to be conducted according to the specific needs of a given industry. So, if you're looking to ascertain what's going on in terms of reliability standards you have to look in three places:

1) The W3C infrastructure initiatives related to XML, the Semantic Web, and security/privacy;

2) Separate industry-oriented consortia such as Rosettanet (a consortium of supply-chain trading partners whose goal is to define the schema necessary to accomplish business trading partner collaborative activities); OBI (Open Buying on the Internet – a consortium of companies dedicated to developing and deploying standards for Internet-based procurement); and the Organization for the Advancement of Structured Information Standards (OASIS) –an international consortium focused on the fostering the adoption of product-independent formats for conducting e-Business. Other industry oriented consortia include Association for Cooperative

Operations Research and Development (ACORD); and Standards for Technology in Automotive Retail; and

3) Additionally, there are cross-industry consortia such as the Business Internet Consortium that consists of members from end-user companies such as, BT Financial Group, Capital One, Charles Schwab, Ford Motor Company, ImageX.com, Pennzoil-Quaker State Company, and Reuters that also influence how Internet standards evolve. This consortium serves as a forum for customers to meet with vendors to discuss emerging technology and business issues with a goal of accelerating the adoption and deployment of next generation e-Business solutions.

Messaging/Routing Overview

There are two bodies of work underway in routing/messaging within the W3C:

1. Work being conducted to enable SOAP-based envelopes to run over protocols other than HTTP (the Internet transport protocol); and

2. Work being done to enable SOAP to be used for one-way messaging, two-way messaging (such as request/response messaging), and for peer-to-peer conversations (where a single message may be passed through a group of intermediaries).

With respect to SOAP over multiple protocols, Bloor Research NA's finding is that work group efforts are progressing well. SOAP is now capable of running over UDP, TCP/IP and other protocol stacks. So, the need for richer, more robust communications protocols has been (and is continuing to be) addressed.

With respect to sending/receiving SOAP messages and passing those messages through multiple intermediaries – this work is important because it helps coordinate information. Information coordination is extremely important – especially when transactions involving the transfer of money or assets are involved. For instance:

A simple Web service may provide a user with the ability to retrieve flight information. This service is a simple "look-up" – a read of a data table and transfer of such information back to a single user.

More complex Web services may involve interactions amongst multiple parties – and also may involve the transfer of money or material. A buyer may place an order with a company that then needs to check its inventory for a product. That product may not be in inventory, necessitating that another program be initiated to ascertain where the product can be found, how much it costs, and when it can be shipped. At this point the simple look-up program becomes an intermediary – acting as an agent for the buyer and seeking to find a product for that buyer. In order to complete this transaction the product must be found and money must be exchanged to purchase goods. Now imagine how many messages needed to be sent to complete this transaction – messages from buyer to first supplier, from first supplier to second supplier and vice versa, and then from the first supplier back to the buyer.

Part of the role of SOAP working groups is to standardize the way that messages are routed, tracked, and coordinated such that transactions such as those described above can be carried out in a reliable fashion.

In short, the standards committees are very active in improving SOAP protocol support and in providing recommendations that address requirements for routing, tracking, and identifying the sources of information as well as intermediaries who had access to that information along its route to a receiver.

Transaction Processing Overview

When you search on transaction processing, reliability, and sometimes on messaging/routing on the W3C.org web site, you invariably end-up at a page with the following heading: *The W3C ECommerce/Micropayment Activity is now Closed!* What does this mean? Has the W3C figured out how to standardize complex e-Commerce transactions; how to handle micropayments (small fees such as pennies or fractions of pennies that will be charged for the use of some Web services applications); or how to ensure that applications run reliably or recover from application/system/network failures? Not so.

Instead, the W3C continues to work on issues related to conducting eCommerce – but has chosen to focus specifically on those related directly to transaction processing/message handling over Web services architecture. As the W3C puts it:

The role of W3C is to focus on core infrastructure technologies for Electronic Commerce and identify common infrastructure needed in this area. W3C is not committed for example in specifying banking systems nor schemas for specific Electronic Commerce applications…

Source: http://www.w3.org/ECommerce/Micropayments/

In short, the W3C appears to be saying: "we will create infrastructure standards (for instance, for routing messages through multiple constituents or for transaction roll-back in case of failure), but we are leaving the task of standardizing content and schema to various industry consortia (which would include OASIS; Open OBI; Rosettanet, and others)".

Manageability Overview

How does one manage distributed applications in a loosely coupled environment? A search on manageability software or solutions on the W3C Web site yields little in the way of manageability efforts underway within the standards setting organization. We found only "key" management" (as in public key encryption – an aspect of Web services security) as we searched the W3C site.

But, a search of vendor sites does yield numerous Web services manageability results. Various vendors offer applications management software products that lend themselves nicely to the management of distributed applications. Numerous security vendors (or vendors that offer security as well as other Web services enhancers) are listed in Chapter 9 of this report.

Performance/Tuning Overview
There are at least four touch points that affect the overall performance of Web services-based applications, including:

- *Application design* (especially in the case of Web services because applications may consist of linked services across multiple, disparate systems environments). The number of cooperative programs that need to be invoked; how well Web services applications are integrated; how well an application is written – all play a distinct role in achieving optimal performance;

- *Overhead* – Web services applications are message-passing intensive. The increased workload of having to process thousands upon thousands of additional messages in order to make Web services work has the potential to bog-down existing server architectures within an enterprise information infrastructure. Additionally, because Web services architecture needs to be enhanced with security, reliability, and other software packages – additional overhead may result. Accordingly, performance tuning must take into consideration the additional processing load related to securing and making Web services reliable, as well as the increased processing load related to message processing.

- *Network characteristics* – the number of hops that must be made to obtain a Web service; network speed/bandwidth characteristics; and other network considerations also play a role in overall performance of Web services applications.

- *System/Storage characteristics* – systems and storage subsystems may also require tuning in order to process Web services applications more quickly. The primary target is systems "latency" (latency refers to the situation where you may have plenty of bandwidth for sending and receiving information over the Internet – but your system may lose time retrieving, processing, or storing data).

In order to optimize Web services performance, all of the above must be examined and tuned to achieve optimal performance.

How does the W3C handle performance and tuning requirements? In short, the W3C Quality Assurance activity concerns itself with, amongst other things, ensuring that the various Web services and XML standards work well together. And in the course of so doing, the Quality Assurance working groups develop test suites and tools (or use tools from other sources) in order to examine the load and scalability implications of standards as they work together. Some of these results are made publicly available (such as the Document Object Model – DOM – Conformance test suite):

The DOM Test Suites (DOM TS) will consist of a number of tests for each level of the DOM specification. The tests will be represented in an XML grammar which ensures that tests can easily be ported from the description format to a number of specific language bindings. This grammar will be specified in XML Schema and DTD form. The grammar will be automatically generated from the DOM specifications themselves, to ensure stability and correctness.

The DOM TS will consist of a number of tests in this XML representation language, plus the XSLT stylesheets necessary to generate the Java and ECMA Script bindings, plus of course the generated code. XSLT stylesheets for other languages will also exist for download but will not form part of the DOM TS Core.

Note: Since this memo was written, this test suite has become publicly available.

Source: http://www.w3.org/DOM/Test/

But, aside from performing a certain amount of in-house testing to ensure interoperability of Web standards, and aside from creating certain test suites that can aid developers in testing application performance, the W3C does not concentrate on building tools or test suites designed to optimize system, network, nor application performance. Instead, this role goes to open source foundations, to IS managers who wish to develop their own testing/tuning scenarios; as well as to vendors who wish to provide such tools and utilities on a product or services basis.

Interoperability Overview

One of the biggest failures of predecessor architectures that sought to promote cross-vendor, cross-platform interoperability was that various vendors interpreted and implemented standards in incompatible ways (causing interoperability issues). Or, sometimes, vendors partially implemented standards – again leading to interoperability issues. In some cases, third-party organizations offered to quality-assure vendor standards implementations – but this process was frequently time-consuming and costly (and oft times third parties were put in the middle of trying to resolve vendor disputes).

To remedy this situation numerous leading vendors have formed the WS-I (Web Services Interoperability Organization) – an organization designed to provide venues for interoperability testing and for "working out issues" between vendor implementations.

Another point to consider with respect to interoperability is the approach used by various vendors to create "adapters" or "connecters" between various legacy applications, custom applications, and packaged applications. These adapters are being written to converse using the Web services WSDL template – thus making it possible for various applications (like PeopleSoft, SAP, Oracle, and more) to communicate readily between themselves – or to communicate to new WSDL applications (such as, for instance, new WSDL decision support applications).

In short, interoperability concerns are being addressed through vendor-at-large interoperability testing as well as through the development of new software adapters that make use of Web services templates such as WSDL.

Vendor/Open Source Findings

Bloor Research NA found a bevy of vendor and open source products that can help enterprises build secure and reliable Web services environments. Part III of this book provides a listing with Bloor commentary on some of these vendor/open source product offerings.

There are dozens of Web services solutions provided by companies such as IBM, CapeClear, Microsoft, Sun, webMethods, SilverStream, Tibco, IONA, Oracle, and others that address Web services shortcomings. (For instance, IBM's Tivoli security products do an excellent job of supplementing its WebSphere product offering with security and manageability features. And the same can be said for products offered by webMethods, Microsoft, and Hewlett Packard – each company offers systems/applications/network management software that can provide manageability features that can be used to manage Web services applications). Other *point products* (individually packaged products) can be purchased from various vendors to address interoperability shortcomings or transaction-handling shortcomings. Still other point-product packages can be purchased to address manageability requirements, routing requirements, security requirements, and the like. See Chapter 9 for more details on these types of products.

.

PART I: A Quick Web Services Primer

What This Section Is About

In order to proceed with a discussion of Web services it is necessary to establish that the authors of this report and the readers are on the "same page" with respect having a common understanding of:

- What Web services are;
- What they are being designed to do; and,
- Where the shortcomings exist in current implementations of the architecture.

Further, it is also important that the reader gain an appreciation of what standards activity is taking place to address current architectural shortcomings, as well as an understanding of the timing for certain Web services enhancements to occur (for instance, when security, routing, or reliability improvements will occur) – in order to best decide when to deploy Web services architecture.

To provide this common ground, this section (Part I), has been designed to provide the reader with a basic understanding of Web services, standardization efforts, and architectural strengths and shortcomings. Additionally, this section includes a discussion on how and why businesses should use Web services (how Web services can be exploited for efficiency gains, competitive edge, new service delivery models, and more).

What We Hope You'll Learn

In this section we hope that you will learn the following:

- Web services are essentially a new program-to-program communications architecture designed and optimized for the Web. If you take only two ideas away after reading this report, make them the following:

1. Web services are distinctly different from predecessor distributed computing architectures because of one key point – *Web services protocols, interfaces, and registry services enable loosely coupled applications to work cooperatively together!* What this means is that predecessor distributed computing architectures have required that applications be "hard-wired" together (tightly coupled) – which means that application developers would have to write links between applications in order for one application to provide a service for another application. Web services enable applications to work cooperatively with any application written in almost any programming language running on almost any platform (provided that both applications can find each other and work out a mutually agreeable way of communicating). *This one factor, this "loose coupledness", changes the whole complexion of how applications can be designed and deployed – and can thereby result in the creation of entirely new business models.*

2. Now, temper the above with the following perspective: *in order for Web services to be used in mission-critical, production computing environments the Web services architecture needs to* *be strengthened in about a dozen areas (including security, routing, transactioning, reliability, and more). Web services standards committees are very actively addressing these shortcomings – but in the meantime, various vendors are plugging these holes with their own products that run on Web services protocols.*

These points are, in a nutshell, what this whole report is about. And they encompass the key messages that are elaborated upon in greater detail in the following chapters.

Other Important Concepts to Understand

In addition to understanding the concepts of loose-coupledness and Web services shortcomings, it is also useful to understand the following perspectives:

- There are currently only two *Web services* standards – Simple Object Access Protocol (SOAP), and Web Services Description Language (WSDL). UDDI (Universal Description, Discovery and Integration) is expected to someday become a standard, but currently is a joint initiative of vendors and businesses.

- XML (eXtensible Markup Language) is also an evolving W3C standard – one that is focused on syntax, data/content, format, schema, and semantics – rather than program-to-program services. Although XML is a crucial element of Web services (because XML represents the way that data and content are packaged to be sent over Web services architecture) – XML is not discussed in depth in this report. The reason: there are almost a dozen working groups focused on various aspects of XML design such as XML Protocol · XML Schema · XML Query · XLink, XPointer, XML Base,· DOM, RDF, CSS, XSL, XHTML, MathML, SMIL, SVG, XML Signature, and canonicalization. It would take a lot of time to explain the intricacies of XML – but the report is supposed to be focused on Web services architectural shortcomings and what to do about them. Hence, XML is covered on a need-to-know basis where relevant specifically to Web services architecture.

- Web services applications are more dynamic than predecessor architectures – meaning they can find other applications and automatically work out how to communicate with those applications. This fluidity aspect is one of the most important aspects of Web services architecture because it enables:

  - Applications to be designed and built in a radically different way (automatically assembling applications using application objects found on the Internet or in an Intranet) – thereby enabling businesses to dynamically and fluidly increase the size of their application portfolio;

  - Business to reduce application development timeframes and costs;

  - Applications to be sold and deployed in a new way;

  - Businesses to respond more quickly to competitive pressure (or conversely, to create competitive pressure by assembling new applications more quickly than ever before);

  - Business to reduce Sales, General, and Administrative (SG&A) costs (when used in conjunction with business process management software) – thus helping to improve organization and individual productivity; and,

  - Developers and integrators solve application interoperability problems.

- Web services are similar to other distributed computing architectures in that most other distributed computing architectures have been designed to functionally accomplish the same thing – provide for cross-platform program-to-program communications. But note that these "predecessor" architectures have already dealt with issues such as security, reliability, manageability, performance, etc. – and that Web services architecture has some catching-up to do. Still, consider that Web services architects can learn from the architectural successes of predecessor architectures – and can mimic those architectures in order to rapidly create a rich, robust set of Web services standards – leading Bloor Research NA to the conclusion that Web services architecture may mature more quickly than predecessor architectures..

- Web services has almost universal industry support (currently over 500 vendor/end user organizations are contributing to the formation of Web services standards). This level of industry commitment makes Web services highly likely-to-be-successful as an industry-wide formal and de facto distributed computing standard.

- Today's Web services have certain "shortcomings" in the areas of:

  - Security/privacy;
  - Messaging/routing
  - Quality-of-services/reliability;
  - Transaction-handling;
  - Manageability;
  - Performance; and,
  - Interoperability.

- The Worldwide Web Consortium (W3C) is actively working to resolve the abovementioned shortcomings – and has an infrastructure and methodology in place to design and develop standards need to overcome these shortcomings. This infrastructure consists of interest groups, working groups, and coordination groups whose purpose it is to define what the standards should look like, make appropriate technical recommendations, and coordinate/communicate those recommendations to other interested standards groups, standard bodies, vendors, and end-user organizations. But the W3C is not the only organization involved in helping build Web services architecture. UDDI.org, Rosettanet, OASIS, WS-I, and dozens of other vendor and business consortia are also contributing to the building of a more robust and secure Web services architecture.

Chapter 1

What Are Web Services?

Getting Started: Common Concepts and Definitions

In order to ensure that the authors of this report and the readers of this report have a common understanding of Web services architecture and Web services standards setting/interoperability testing, the next four subsections examine:

1. What Web services are;
2. What they do;
3. How Web services standards are set; and
4. How they enable cross-platform program-to-program inter-operability to be achieved.

What Are Web services? — Quick Definition

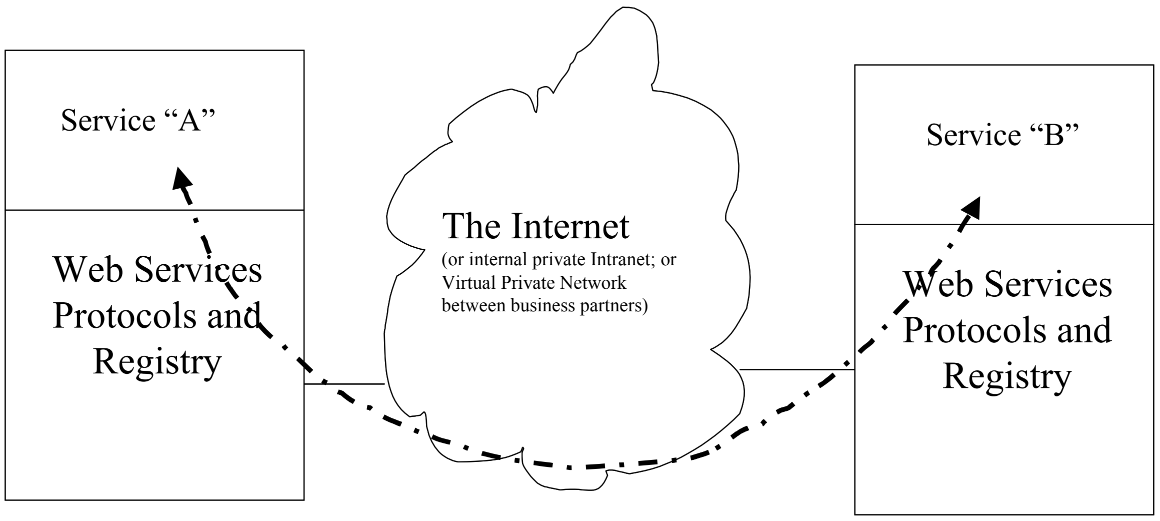
Web services are a series of standards and evolving standards that are being designed and specified by the Worldwide Web Consortium (W3C) to foster cross-platform program-to-program communications. More specifically, the W3C has currently specified a template and a procedure call protocol as "official" Web services standards (recommendations). They are Simple Object Access Protocol (SOAP), and Web Services Description Language (WSDL). Universal Description, Discovery, and Integration (UDDI) is an evolving standard (part of an independent initiative – UDDI.org).

What are Web services – A Longer Definition

Web services are a series of standards and evolving standards that are being designed and specified by the Worldwide Web Consortium to foster cross-platform program-to-program communications (see Figure 5).

Translation: The W3C (a standards organization) is formulating Internet computing standards that enable applications written on one systems platform (irrespective of what operating environment or programming language was used) to communicate with other applications on similar or disparate systems architectures. During this program–to-program communications, one application performs a service (a data look-up, a file transfer, a calculation, or other programmatic function) for the requester application. "Web services" is the name for this evolving series of application service standards.

Figure 5 – Web Services: The Basic Concept


Service "A"
Web Services Protocols and Registry
The Internet
(or internal private Intranet; or Virtual Private Network between business partners)
Service "B"
Web Services Protocols and Registry

Source: Bloor Research North America – May 2002

More Specifically…
More specifically, the W3C has currently specified SOAP (a remote procedure call method for invoking applications) and WSDL (a template/interface for defining how applications can "talk" to each other) as official specifications. UDDI (a registry/directory evolving standard) is not an official W3C standard – but is expected to become a standard over time.

At Bloor Research NA, we describe Web services protocols and the related registry service as follows:

- ***Integration*** — SOAP is used for exchanging information in a loosely coupled environment. If you are familiar with communications APIs – application program interfaces; or RPCs – remote procedure calls – you'll understand what SOAP does. If not, suffice it to say that SOAP is a set of action words (verbs) that enable applications to invoke each other, to connect, and to communicate with each other.

- ***Behavior*** — WSDL is like a map or template. It is used to describe the way that applications can communicate with each other. For instance – one application may wish to format a message in a particular way while a potential cooperative application may expect it formatted in a different manner. WSDL gets each of these applications to agree on common ways to present data types such that information can be shared between applications. WSDL also coordinates;

    - Messages;

    - Operations;

  - Port types (and Ports themselves) to be used for communications;
  - Binding (communications protocols); and
  - Services.
- ***Location*** — UDDI is an evolving registry standard (that can make use of WSDL and SOAP). It is a universal directory-like repository that contains information about various Web services applications.

Other Important "Elements": XML and HTTP

The preceding definitions establish Web services standards as protocols for exchanging information, as well as a standard registry service for finding and sharing applications and services. These definitions also refer to "XML" (eXtensible Markup Language). A simple definition for XML is that XML makes it possible to format data such that different applications can read it. A more detailed description would describe XML as a human and machine readable "meta-language" that helps to describe content and data that is to be shared between applications. XML provides a means to present data, syntax, schema, and semantics when sending a file – far more than its predecessor, HTML (hypertext markup language) that primarily allows for presentation-only of content and graphics.

What does all of this boil down to? In short XML allows data and content to be packaged in a *common format* that is readable and is able to be manipulated between similar or disparate application environments. It's like a letter – one that all recipients understand how to read, and from which elements can be cut, pasted, manipulated, mixed, matched and used to create other documents or dropped into other forms for data processing. XML can carry content (written information such as reports or letters) – or it can carry data (such as transactional information or numerical information from databases).

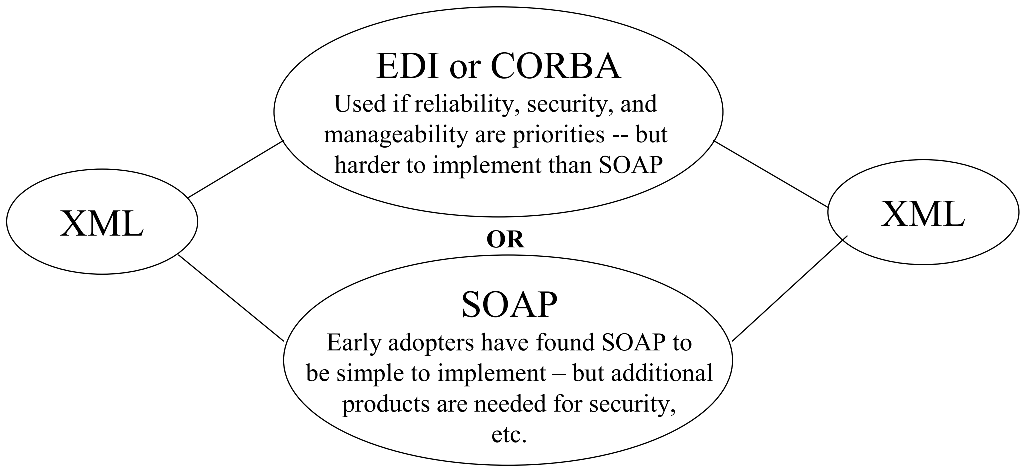
HTTP is a communications protocol. It's used to transport data across the Internet. Note: today most Web services SOAP messages are sent using HTTP. But SOAP is transport independent meaning that it could be used over other transport protocols besides HTTP to open sessions and send/receive data.

How Do Web Services Work?

How do all of these protocols and the registry service work in concert? Web services architecture is designed to allow applications to be published in a UDDI directory where other "requestor" applications seek out their services. WSDL will be used negotiate how to communicate (what format is to be used, what ports, which communications protocols, etc.). And SOAP will provide a programmatic interface that makes it possible for applications to "talk" to each other.

Notice that the future tense is used in the preceding paragraph. This is because today Web services architecture is primarily being used to pass XML (extensible markup language) data and content between server environments using the SOAP programmatic interface over HTTP (see Figure 7).

Figure 7 – How Web Services Architecture Is Really Being Used Today


EDI or CORBA
Used if reliability, security, and manageability are priorities -- but harder to implement than SOAP
XML
XML
OR
SOAP
Early adopters have found SOAP to be simple to implement – but additional products are needed for security, etc.

Source: Bloor Research NA, May 2002

For enterprises that wish for more robust protocols that have stronger security and reliability, XML can be passed over architectures like CORBA (Common Object Request Broker) or EDI (electronic data interchange). Bloor Research NA expects the use of WSDL and UDDI to increase greatly over the next five years as enterprises first use UDDI to construct internal libraries of reusable application modules; and then second, as Independent Software Vendors (ISVs) start to publish their products in public UDDI registries (such as those operated by Microsoft, IBM, HP, NTT and SAP).

Chapter Summary

There are two Web services standards (SOAP and WSDL). These standards are used to invoke other applications (for program-to-program communications) and for describing how applications can work together (over which ports, sending what types of information, using which protocols, etc.). UDDI adds a dynamic aspect to Web services because it enables applications to automatically find each other and work together.

XML is also a W3C standard – but it is not a “Web services” standard per se. Instead, XML is a formatting/packaging standard for content; Web services are architectural standards for communicating between programs. Note that XML data and documents can be sent using other distributed computing architectures such as CORBA or EDI – illustrating that XML is not solely designed for use with the W3C Web services architecture.

HTTP is a simple transport protocol (a line protocol) used heavily for Internet communications.

How do these protocols and the registry services work in concert? XML provides a common format for packaging data and documents (as well as graphics). SOAP, WSDL, and HTTP work together to enable cooperating applications to communicate and negotiate with each other. UDDI provides a listing of available Web services as well as other additional information that enables cooperative applications to understand how to work with listed Web services applications.

Chapter
2

Why Do We Need Web Services?

Webservices.org – a non-profit organization that contributes to the overall Web services knowledge base and that influences the Web services standards committees – does a stellar job in articulating why Web services are so important. According to Webservices.org, the key benefits of Web services relate to:

Software as a Service - As opposed to packaged products, Web Services can be delivered and paid for as streams of services and allow ubiquitous access from any platform. Web services allow for encapsulation. Components can be isolated such that only the business-level services are exposed. This results in decoupling between components and more stable and flexible systems.

Dynamic Business Interoperability - New business partnerships can be constructed dynamically and automatically since Web Services ensure complete interoperability between systems

Accessibility - Business services can be completely decentralized and distributed over the Internet and accessed by a wide variety of communications devices.

Efficiencies - Businesses can be released from the burden of complex, slow and expensive software development and focus instead on value added and mission critical tasks . Web services constructed from applications meant for internal use can be easily exposed for external use without changing code. Incremental development using Web services is natural and easy and since Web Services are declared and implemented in a human readable format there is easier bug tracking and fixing. The overall result is risk reduction and more efficient deployability.

Universally - Agreed-to Specifications Web Services are based on universally agree specifications for structured data exchange, messaging, discovery of services, interface description, and business process orchestration.

Legacy - Integration Greater agility and flexibility from increased integration between legacy systemsNew Market Opportunities There will be greater feasibility to the dynamic enterprise and dynamic value chain businesses.

Source: Webservices.org Web Site:http://www.webservices.org/index.php/article/articleview/75

The remainder of this section takes a closer look at each of these benefit statements. But before doing so, it is important to understand two concepts: application componentization (using "objects") and encapsulation.

Before Discussing Benefits – One Important Concept: Application Componentization and Encapsulation

In the olden days of writing application code, applications were designed in a monolithic fashion (all the code that enabled an application to work was tied together in large blocks with the application). Note that:

- If the code was sold commercially, this monolithic approach one oft-times resulted in the creation of large software packages that were difficult to deploy and maintain – and that frequently required customization to address the specific business or industry needs of the buyer.

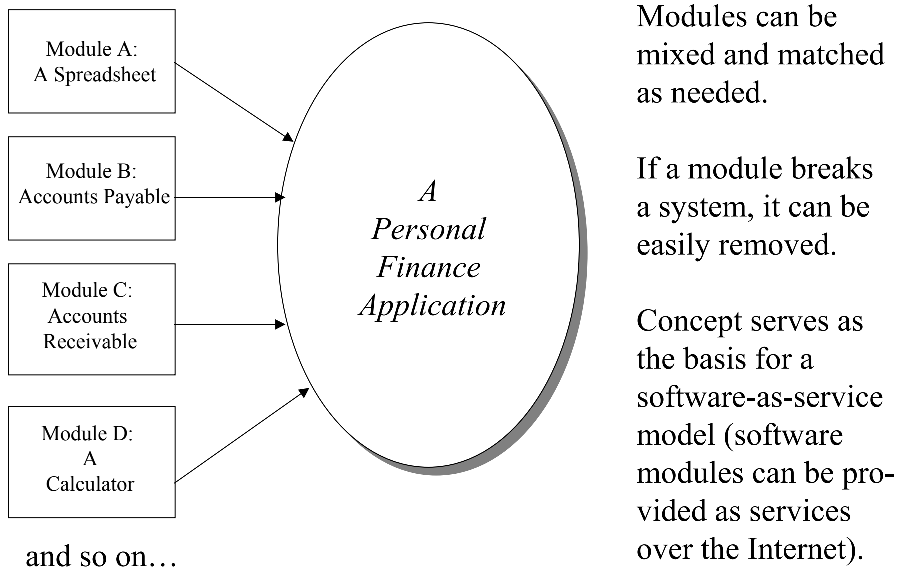
- If the organization created its own custom code, this monolithic approach resulted in the creation of one big chunk of code that needed to be constantly updated and maintained.

In order to expand application functionality, application modules were "bolted-on" to the original code – sometimes introducing software bugs or leading to software conflicts with various parts of the application monolith. A software change that added new functionality could often break some existing features that resided in the original code block. Creating large packaged or custom applications, therefore, required tremendous amounts of quality assurance in order to ensure that new application modules that were attached to the existing core code did not break the existing code set.

Over the past several years applications development has been shifting to the creation of application "objects" (modularized application components that can be mixed-and-matched to create entire applications). These components can be encapsulated (made discrete) and can then be linked together to create larger applications. See Figure 8 – an illustration of how various modules can be linked together to create a personal money management application.

This concept of creating application modules (also called "objects") is important to understand because one of the major advantages of Web services architecture is its ability to link application modules together to create more complex, highly-functional applications.

Figure 8 – Linking Application Objects to Create a Large Functional Application


Module A:
A Spreadsheet
Module B:
Accounts Payable
Module C:
Accounts
Receivable
Module D:
A
Calculator
and so on…
A
Personal
Finance
Application
Modules can be mixed and matched as needed.
If a module breaks a system, it can be easily removed.
Concept serves as the basis for a software-as-service model (software modules can be provided as services over the Internet).

Source: Bloor Research North America – May, 2002

Now that you have a basic understanding of how application object components can be created, encapsulated, and then combined to create new applications, the following benefits that can be derived using Web services should make a lot more sense.

Software as a Service

As described in Figure 8, discrete application modules can be created – and those modular components can be combined with other modules to create larger and more functional applications (in the example the larger application is a unified, integrated personal finance application). In this example, modules in the same server environment were encapsulated to form an integrated application.

Now imagine this: there is no reason that all of these modules have to be co-located on the same server. Using Web services architecture, application modules can be "packaged" for delivery over the Internet to customers who desire to use them. These packages can consist of:

- New software that is to be permanently deployed on a receiving system;
- Software upgrades to existing code;
- Software for one-time use; and/or
- Software to be used as a service on a per use, as needed, or ongoing flat fee basis.

The first two cases involve using the Internet for software distribution. But the latter two cases show how many vendors are planning to use the Internet to deliver software-as-a-service.

This software-as-a-service concept is revolutionary. In short it changes the way that applications can be delivered and paid for (as streams of services). For vendors, new software products could be delivered to market more efficiently (and an opportunity to charge service fees would also be prevented). Further, vendors would be able to keep they installed bases up-to-date on the latest software revisions, potentially reducing software development costs. Enterprises would be able to pay for only the software they used (rather than having to buy complete software bundles with components that they may or may not use). And enterprises would have access to a wealth of application services from a myriad of vendors (potentially leveling the playing field between large software makers and smaller vendors – and maybe even driving down prices for software services accordingly). This new approach has the potential to change the way that user organizations purchase and use software; the way the vendors market software; and even create a new class of software sold by end user organizations to the marketplace in general – all as application services!

Dynamic Business Interoperability

Webservices.org describes this benefit as the ability of business partners to dynamically form partnerships by combining their respective application bases. What Webservices.org is talking about is the concept that Web services applications don't care about what operating system, or platform, or programming language a cooperative application is based on. Web services enable disparate applications to communicate and share data – and by so doing lend themselves nicely to helping to overcome system-to-system, application-to-application interoperability issues. (Note: this interoperability would be achievable in a perfect world where each application developer adheres to standards properly – a world that the Web services Interoperability Organization is hoping to realize).

Imagine if this kind of instant interoperability were achievable! Businesses would spend less time (and money) working out cross-platform systems, application interoperability, an integration issues. Instead, businesses would be able to concentrate on how to penetrate new markets; or concentrate on how to forestall competition; or focus on how to better service customers. And if businesses could get their strategic foci in line, they could form partnerships with other vendors to fill-in-the-gaps within their respective product lines (thus forestalling competition, or creating new competitive pressure). And ultimately, these new, fluid partnerships would result in the delivery of richer application services to customers.

Because Web services allow for this type of dynamic mixing-and-matching of applications, dynamic business interoperability is achievable. But also, don't fail to notice that this dynamic business interoperability changes the way that businesses can compete, as well as changing the nature of partner relationships – resulting in the creation of new business models involving new dynamic partnerships. Web services have the potential to fundamentally change the way that businesses organize and partner in the future.

Accessibility

Because Web services make use of the Internet, the source of any application service can be located wherever a processing device can be connected to the Internet. This means that providers of application services can be located in Gnome Alaska, or Pretoria, South Africa, or anywhere in the world where an Internet connection is available – and from those locations provide services anywhere in the world that those services are required (as long as the requester has Internet access).

So imagine that a small, two-person business in New Delhi, India makes its Web service available on the Web. That shop would suddenly have worldwide exposure for its product. And again, because of the way that Web services work, the traditional barriers to market entry (such as manufacturing, distribution, and advertising) can be eliminated – resulting in the creation of a new go-to-market business model for small software entrepreneurs (and this model also applies to mid-sized and large software companies).

In addition to enabling vendors to reach-out and deliver services anywhere in the world, Web services also enables geographically distributed and disparate devices to reach-in and obtain services from anywhere an Internet connection is available. In short, Web services can enable global accessibility to services from a plethora of different device types provided those devices can establish an Internet connection.

Efficiencies

Web services provide the opportunity to improve operational efficiency from at least two angles:

1) Reduced software development time/effort/cost; and

2) Reduced Sales, General, and Administrative (SG&A) costs when used in conjunction with business process management software.

First, from a reduced cost for software development perspective, Web services enable application developers to construct applications on-the-fly by assembling application objects into full-fledged and powerful business applications. As a result, the speed at which applications can be designed and built increases while development schedules are condensed. Applications that may not have been feasible to build due to probable development time and cost using traditional methods often become “suddenly feasible” using Web services (because the development effort moves from creating core modules to weaving together existing modules to form a solution). Other cost saving factors also exist: only one interface to learn, no OS specific code to write, and, most importantly, development tools, like WSAD, do most of the work. As a result, developmental efficiency increases (and enterprises are often able to bring new initiatives to market – thus increasing new business opportunities).

In the second case, (when Web services architecture is used with business process management software), great organizational efficiencies can be realized. Imagine a scenario where an organization is able to use Web services components to streamline business processes. For instance, a new way of processing time and expense (T&E) reports could be introduced, eliminating an existing paper-based reporting system. That organization could use a simple spread sheet for inputting time, link that spreadsheet to a workflow program, and track it through the system. That spread sheet could automatically linked to a customer invoicing system (if it were a billable expense), eliminating the need for an administrator to manually input it into a billing system. And executive management would be better able to track expenses, while the employee who originally submitted the bill would always know where the bill was in the "system" such that the employee could determine when reimbursement would occur. All of this could be accomplished by linking applications together using Web services and overlaying those applications on a workflow (business process management) system. The end result: improved organizational efficiency and reduced costs.

Universally Agreed-to Specifications

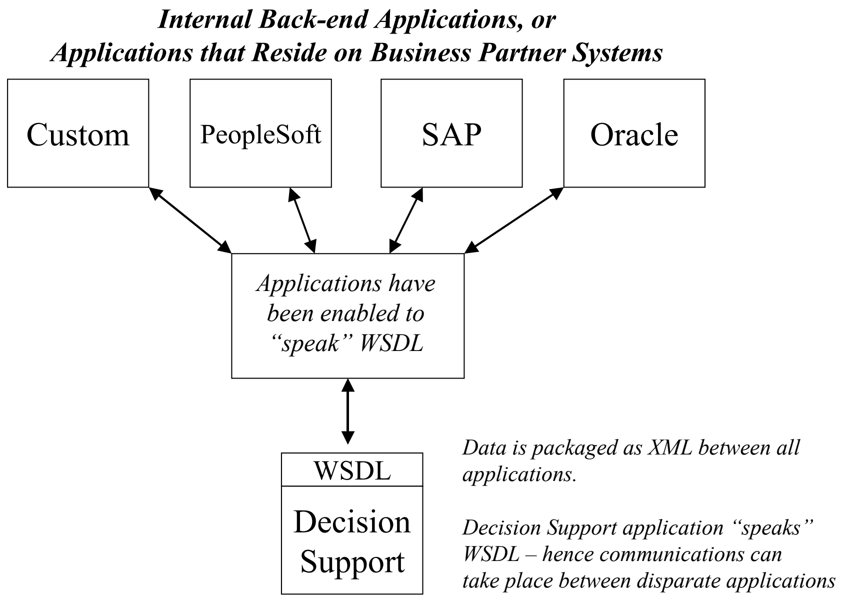
According to Webservices.org "Web services are based on universally agree specifications for structured data exchange, messaging, discovery of services, interface description, and business process orchestration". This translates into the following benefit: organizations that are able to use technology standards can lower their product and application acquisition and implementation costs by reducing the amount of custom integration work that needs to occur to integrate products. Additionally, proprietary solutions often cost more than similar standards-based solutions (because only one company competes when providing a proprietary solution, whereas multiple companies compete when standards approaches are used – thus driving down costs). The bottom line is that standards tend to drive down total cost of ownership (TCO) – an important consideration for most enterprises.

Legacy Integration

Imagine the following scenario: your organization has several different packaged and custom programs (human resources, financials, sales and distribution, etc.) – all of which contain important data that needs to be analyzed. The big problem that you face is that none of these applications package and share data in the same way. So how do you obtain run-the-business data from these applications in order to compile a unified report on the state of health of your business?

Web services can be used to help overcome application integration issues. Here an example of how (see Figure 9):

Figure 9 – Integration of Legacy Data Using A Web Services-enabled Decision Support Application


Internal Back-end Applications, or
Applications that Reside on Business Partner Systems
Custom
PeopleSoft
SAP
Oracle
Applications have been enabled to "speak" WSDL
WSDL
Decision Support
Data is packaged as XML between all applications.
Decision Support application "speaks" WSDL – hence communications can take place between disparate applications

Business Result: ***Information from disparate systems can automatically be shared. Reduces specialized programming requirements and stream-lines administrative data acquisition and reporting tasks.***

Source: Bloor Research North America – May, 2002

In Figure 9, all applications are capable of presenting data in XML format (meaning that data is available in a common format between the decision support application and the internal back-end applications). The next step is to work on the "connectors" or "adapters" between the business applications and the decision support application. By using WSDL, data can be accessed and shared without necessitating a lot of custom programming to proprietary interfaces. As a result – applications can in many cases be easily connected and data can be readily shared – overcoming huge and costly barriers to application interoperability. And thus, interoperability between legacy applications and new applications can be achieved using Web services protocols.

New Market Opportunities

As described in previous sections, Web services have the potential to change the way that application programs are developed; how software makers can bring products to market; how competitors can create and respond to competitive pressure; and much, much more. As a result, businesses will be able to rapidly create new applications; create new partnerships; more easily deliver products and services to market; and more easily integrate existing enterprise applications with those of supply chain business partners.

As Web services-based applications proliferate, whole new business models will evolve – opening-up new business and market opportunities. And when combined with business process engineering software, Web services will also help streamline business operations – resulting in increased efficiency and lowering the general costs associated with administering and operating a business.

Web services will have a very profound affect on how enterprises structure their businesses as well as information systems to compete and open new markets in the future.

Chapter Summary

One of the most important concepts to understand about Web services is application modularization/componentization. The creation of self-contained application objects make it possible to build applications by assembling application "services" from a library of application components. In essence, a series of application objects can be connected using a building-block approach to create complex and robust applications.

If you understand this basic concept, then the real eye-opener is what business benefits can be derived using this approach. By being able to create applications on-the-fly:

- Enterprises can reduce their application development costs, eliminate application interoperability issues; more easily establish communications and information-sharing systems with their business partners; and a whole lot more.

- Enterprises can also build new applications more quickly (increasing the opportunity to sell new products or open new markets). And, when combining Web services and business process management software, enterprises have the potential to streamline business operations, resulting in reduced SG&A costs that pass directly to the enterprise bottom-line in terms of increased profitability.

- Software vendors can deliver their products to market more effectively (using Web services for software distribution and updating); and software vendors can also bring new products to market more effectively and at less cost (simply listing a product in a UDDI registry has the potential to give a vendor worldwide presence at virtually no cost). Dwell on this concept a bit and you'll realize that the barriers to entry for many start-up software firms can easily be overcome using Web services.

- Enterprises will be able to lower costs for integrating applications internally (such as integrating human resource systems with financial systems, or sales with manufacturing and distribution); as well as externally with corresponding order processing and financial systems of their business partners. Web services have the potential to save millions of dollars in application integration costs annually.

The bottom line with Web services is that they have the potential to radically influence how businesses use information system. They can enliven new business models, streamline supply chain interaction, reduce development costs, open new business opportunities, and much, much more. And this is why it's so important for business executives to understand what Web services are, what benefits they offer, and where shortcomings and limitations exist in today's Web services architecture.

Chapter
3

Web services Gotchas According to Bloor Research NA

So far, this report has focused on explaining what Web services are and what benefits can be derived using them. And the tone has been very optimistic… Now, prepare to switch gears: despite the great promise that Web services holds for improving efficiency and broadening application portfolios, there are years worth of additional standards work that must be done to augment the Web services architecture order to enable Web services applications to be run in production, mission-critical computing environments using formal standards. This chapter sets the stage for the rest of the report by identifying the primary issues faced as standards committees look to make Web services robust, reliable, secure, manageable, and capable of being used in challenging, mission-critical production computing environments of the future.

Where Do Web Services "Need Improvement"?

One of the purposes of this report is to provide business and IS executives with an understanding of the weak points, shortcomings, or gotchas that exist in Web services architecture as it currently stands today. To this end, Bloor Research NA has identified seven shortcomings that are in the process of being addressed through various standardization efforts as well as by consortia, vendor, and open source communities. Those shortcomings are:

- Security/privacy;
- Messaging/routing;
- Quality-of-services/reliability;
- Transaction-handling;
- Management;
- Performance; and,
- Interoperability.

What Needs to Be Improved?

Succinctly, we believe that Web services architecture needs the following improvements in order to be made enterprise-ready:

- **Security/privacy** – We consider Web services architectural security as consisting of two levels: network-level security and content level security.

  At the network level we see a layer of security protection existing at the "line" level with Secure Sockets Layer (SSL) security already being implemented by numerous enterprises to protect data and content in-flight across the Internet. (SSL enables data to be encrypted). But we also see a need for plugging-the-hole that is created when Web services use the HTTP transport protocol to "tunnel" through enterprise firewalls in order to interoperate with certain applications. To fix this situation we note:

  - That SOAP protocol can be driven over other transports – we'd like to see the simplicity aspect of driving Web services over HTTP echoed over other more robust transport protocols;
  - That IBM, Microsoft, and Verisign have collaborated on a series of specifications (WS-Security) that help provide additional security for SOAP messages).

  Both of these activities should do much to help plug-in the holes in Web services line/protocol-level security.

  On the content security side, the W3C has put much effort into securing XML content. Standards recommendations exist for many of the following security/privacy considerations:

  - Protect private data/document *confidentiality*;
  - *Authenticate* where data/content has originated and *validate* its origin;
  - Provide only *authorized* users with access to certain types of content;
  - Ensure the *data and content integrity* that has been sent between communicating entities; and,
  - Provide for *non-repudiation* (a record that shows what transpired and who/what initiated it such that a transaction can be traced along its route and no aspect of the transaction can be denied).

  From Bloor Research NA's perspective, to make Web services secure at the content level we need to see the abovementioned standards recommendations implemented in various vendor or open source products and then deployed appropriately in an enterprise environment. (We currently see a lot of security protection at the line level – but not enough security protection at the content level – within most enterprises today).

- **Messaging/routing, Reliability/quality-of-service, and Transaction Processing** – (All three of these are intertwined). One of the first things to consider with respect to messaging/routing is that Web services architecture is a "message-based" architecture. What this means is that Web services applications need to send a lot of messages back-and-forth in order to request, provide, and receive various services. It is therefore extremely important that messaging/routing be efficient – and that systems administrators are able to track and understand what is happening with messages should a failure occur. The primary issues to be solved with respect to messaging/routing is that ways must be found to track one-way messaging, two-way messaging (such as request/response messaging), and for peer-to-peer conversations (where a single message may be passed through a group of intermediaries). At issue is creating a trail for simple messaging purposes – as well as creating a transaction trail for complex transaction roll-back in case of transactional failure (see next subsection for additional details on roll-back).

  The bottom-line: messaging/routing services need to become more sophisticated in order to allow for greater reliability in transaction processing as well as for audit trails of messages.

  Transaction-handling in particular: In order to be successful in business environments it is extremely important that Web services architecture provide certain transactional "guarantees". Specifically, Web services architecture needs to be able to provide a means for various applications to interact and message with each other – and to recover should some form of technical or process failure occur. The challenge at hand is to ensure that complex transactions are handled in a highly-reliable manner – and if failure should occur, transactions should be capable of "rolling-back" processing to the original, pre-request state.

  In order to address this requirement for messaging reliability, the W3C has stated that its role is to focus on core infrastructure technologies for Electronic Commerce and identify common infrastructure needed in this area. W3C is not committed for example to specify banking schemas for specific Electronic Commerce applications… What this basically translates into is that the W3C will seek to fortify reliable messaging focused around the SOAP protocol.

  What this does not translate into is the creation of a "transaction monitor". In this case, a transaction monitor is a program (or suite of programs) that help manage transaction processing environments. These monitors provide systems administrators with tools to track and tune application performance, to monitor and manage application behavior, and otherwise assist in ensuring that transactions can be processed reliably while performing well at the same time. The W3C is clearly stating that it will work the protocol level – and that means that if IS buyers want products that help manage and tune their Web services-based transactional environments, they may need to turn elsewhere.

In short, the W3C has left the role of providing basic schema (formatting of forms) to consortia such as OASIS; Open OBI; Rosettanet, or to vendors who have an interest in creating transaction monitors for Web services environments. From Bloor's perspective this is fine – vendors such as BEA, HP, and IBM have plenty of experience creating such software environments; and some consortia have also devised transaction monitors that can be run on Web services architecture (for instance, the OASIS Group has specified a Business Transaction Protocol (BTP) for complex transaction processing/transaction monitoring.

Transaction processing is one of the primary uses for computing technology today – and without satisfactory transaction-handling capabilities, businesses will not widely adopt Web services architecture. It is for this reason that Bloor Research NA has identified transaction processing as a critical element that must be addressed in Web services architecture in order for this architecture to be considered mission-critical capable.

- **Manageabiity** — In order to reliably and efficiently operate a distributed computing environment, systems administrators need programs, tools and utilities that provide them insights into the health of their systems and networks, and into the status and behavior patterns of their applications. Although such programs/tools/utilities have existed for years in traditional distributed computing environments, equivalent (in terms of sophistication) programs/tools/utilities do not yet exist for the even-more-complex world of loosely coupled applications.

  As is the case with transaction processing, the W3C is most concerned with protocol and infrastructure design and specification. To this end, the W3C has focused to date on public key management (the management of security keys that are exchanged over the Internet in order to authenticate sending/receiving parties and allow for data/content to be passed). This being the case, it is not likely in the near term that the W3C will focus on the management of loosely coupled applications. Instead, this role has become (and is becoming) the domain of vendors who already provide systems/network/application management frameworks (such as IBM with Tivoli; HP with OpenView; Computer Associates with Unicenter; et al).

- **Performance/Tuning** – As is the case in transaction processing and network/systems/application management, the W3C has chosen to stay focused on protocols and infrastructure – and does not focus on creating specifications for tuning tools and utilities for distributed Web services applications and servers. Having said this, it must be noted that the W3C Quality Assurance activity concerns itself with, amongst other things, ensuring that the various Web services and XML standards work well together. And in the course of so doing, the Quality Assurance working groups develop test suites and tools (or use tools from other sources) in order to examine the load and scalability implications of standards as they work together. Some of these results are made publicly available.

But, from Bloor Research NA's perspective, more and richer programs/tools/utilities need to be developed to optimize the performance of loosely coupled applications in a distributed Web services environment. People are only willing to wait a matter of seconds for a transaction to complete – not minutes, and certainly not hours. Without tools that can make Web services applications perform like tightly coupled applications, Web services architecture will not succeed in supply chain and customer service segments of the computing industry. And without success there, Web services will not succeed in the business applications market segment in general.

- **Interoperability** – For Web services architecture to realize its full potential (as an architecture for cross-platform program-to-program communications), interoperability between various vendor's platforms and Web services implementations must be assured. And achieving interoperability is not a short-term thing – as new W3C specifications and recommendations are released and as new vendors enter the market, more and more interoperability testing will need to transpire between broader and broader mixes of products. Interoperability testing today is fairly straightforward (testing of XML documents over SOAP, sometimes with WSDL mixed in). But in the long term as more and more Web services standards are developed and more and more vendors look to certify their implementation against each other, Web services interoperability testing has the potential to be an immense, costly, and time consuming effort. In other words, Web services interoperability testing is a long term, potentially time consuming effort – and is thus an exposure to companies looking to implement Web services architecture over time.

  In the past various vendors, independent third parties, and even standards organizations have stepped forward to offer interoperability testing and validation services. These approaches were all well-intentioned – but frequently ran into other issues (such as vendors being suspicious of other vendors; funding; lack of resources to carry out testing in an adequate timeframe; and lack of funds to make interoperability testing self-sufficient). As a result, the responsibility for interoperability testing has often fallen on IS buyers or on systems integrators.

  Note that the W3C has not signed-up to provide broad scale testing of various vendor's implementation of W3C standards. (It does however test its own standards recommendations internally in order to assure that one standard recommendation does not conflict with another recommendation). Instead of creating a W3C Interoperability Testing Lab service, the W3C is relying on an industry consortium called the Web Services Interoperability Organization (the WS-I) to perform such tests.

This reliance on a third party consortium constitutes an exposure to all of Web services. Already the WS-I can be accused of some political vendor positioning (as Sun Microsystems, the company that founded the Java programming language was not invited until a few days before the announcement of the WS-I to become a member). And who is to say that the WS-I will be able to achieve its interoperability testing goals for the life of Web services architecture. Because the WS-I is so young, and because we know that the WS-I will face political issues as well as potential long-term testing of complex architecture issues, we at Bloor Research NA see interoperability as a shortcoming of Web services architecture.

Another Opinion

Bloor is not alone in many of its observations about Web services shortcomings. IBM Corporation, on its DeveloperWorks web site, has an article written by a third party that identifies several similar shortcomings as well as other "considerations" that Web services standards committees will need to evaluate as they look to create effective standards for various aspects of Web services architecture. Some of these include:

Discovery. How does a Web service advertise itself for discovery by other services? What happens if the service changes or moves after it has been advertised? WSDL (Web Services Definition Language) and UDDI (Universal Description, Discovery and Integration) are two new standards that address this issue.

Reliability. Some Web service hosts will be more reliable than others. How can this reliability be measured and communicated? What happens when a Web service host goes off-line temporarily? Do you locate and use an alternative service hosted by a different vendor, or do you wait around for the original one to return? How do you know which other vendors to trust?

Security. Some Web services will be publicly available and unsecured, but most business-related services will use encrypted communications with authentication. It is likely that HTTP over SSL will provide basic security, but individual services will need a higher level of granularity. How does a Web service authenticate users? Do services need to be able to provide security at the method level? If you sign up with a vendor that provides services around the world, how do these services learn about your security privileges?

Transactions. Traditional transaction systems use a two-phase commit approach — all of the participating resources are gathered and locked until the entire transaction can take place, at which point, the resources are finally released. This approach works fine in a closed environment where transactions are short-lived, but doesn't work well in an open environment where transactions can span hours or even days. Microsoft supports an alternative scheme, called compensating transactions, in their new XLANG system for distributed business processes. Should this kind of transaction be integrated into Web services? If so, what is the overlap between this approach and proposed standards like XAML, an upcoming XML markup language for supporting traditional transactions?

Scalability. Since it is possible to expose existing component systems like Enterprise Java Beans as Web services, it should be possible to leverage the load-balancing and other scalability mechanisms that already exist. But are there unforeseen stumbling blocks along this path? Does there need to be a new kind of "Web service" application server?

Manageability. What kinds of mechanisms are required for managing a highly distributed system? Since the properties of the system are a function of the properties of its parts, do the managers of each of the various Web services need to coordinate in a particular way? Is it possible to "outsource" the management of some Web services to other Web services?

Accountability. How do you define how long a user can access and execute a Web service? How do you charge for Web services? Will the dominant model be subscription-based, or pay-as-you-go? If you sell a Web service, how do you designate the ownership has changed? Can a Web service be totally consumed on use, or can you reuse the service multiple times as part of your purchase agreement?

Testing. When a system is comprised of many Web services whose location and qualities are potentially dynamic, testing and debugging takes on a whole new dimension! How do you achieve predictable response times? How do you debug Web services that can come from different vendors, hosted in different environments and on different operating systems?

Source: IBM's DeveloperWorks Web Site:
http://www-106.IBM.com/developerworks/webservices/library/ws-peer1.html:devzone=ws

It should be noted that this article does not necessarily represent IBM's actual point of view (it was written by a third party). But it is important to Notice how this IBM opinion echoes some of the major concerns that Bloor Research NA has raised – and adds a few others. More specifically, this IBM document raises concerns about *discovery, scalability*, and *accountability*. We strongly agree with IBM that these issues are important considerations for the future of Web services. But we don't see necessarily see these issues as absolute requirements for making Web services architecture enterprise-robust.

Still, let's take a closer look at these three particular issues. With respect to *discovery*, the point that the author of this report is making is that Web services applications need to be able to find or "discover" each other. The way that this will be done is that applications "publish" their characteristics in application repositories (UDDI repositories). When an application seeks to discover a working partner application, it queries the UDDI registry and uses WSDL to negotiate the terms under which the two applications will interoperate. Discovery enables applications to automatically find each other and work together – but discovery relies heavily on the widespread availability of public and/or private UDDI registries (and these registries are not expected to become highly populated for several years). In the meantime, Bloor Research NA's research shows that enterprise-class Web services architecture can be built without using UDDI registries (simply by sending XML over SOAP, or using WSDL with custom connectors). Hence, we agree with the IBM author that discovery will be important to the future of Web services – but we observe that it will take years for UDDI registries to become populated and we think that enterprise-class Web services environments can be architected in the meantime without UDDI services.

Perhaps another point to be raised with respect to UDDI registries is that other research analyst firms expect that *private* UDDI directories will thrive before *public* UDDI directories. The concept to be raised here is that large organizations and ISVs are likely to have hundreds of reusable application modules. These modules can be put into a common *internal library* and shared by all internal resources that meet the proper authorization requirements. These internal private UDDI registries are far simpler to architect and maintain than public UDDI registries because they are under the control of an internal Information Systems organization. Public directories face larger hurdles in being established because they would need to take applications from a variety of sources; host those applications; potentially assure their quality; find ways to collect revenue from those applications; and would potentially have to face a lot of legal issues (such as protection from liability should a Web services application contain a virus…).

With respect to scalability, the point being raised by the IBM author is that Web services architecture may be able to leverage the processing power of multiple application servers – making it potentially possible to process large applications more quickly by spreading the load of processing across many servers. At this juncture in Web services architectural development, this issue is not of huge concern to Bloor Research NA – but we agree that this idea is intriguing.

With respect to accountability, the IBM author touches on many important considerations. The first issue raised – how does one define how long a user can access and use a Web service – is an important consideration because of potential abuse of service privileges. Imagine a scenario where one competitor accesses a competitor's service and floods it in order to create a denial of service situation. We, again, agree with this author – over time, ways will need to be found to define how services can be used fairly and efficiently – but we also think that it will take a few years for the software-as-a-service model to catch on. And again, in the meantime we still think enterprise-robust Web services architecture can be developed without this level of accountability.

Other accountability scenarios have to do with how to charge for services and what the usage model for services should be. Further, if you pay-as-you go, how does the service provider track and prove that you used the service. Again, these are important issues – but at this juncture this "accounting for services provided" issue is a bit premature considering that UDDI discovery, micro-transaction processing, reliability, and other issues have not yet been satisfactorily resolved.

In summary, we strongly agree with this IBM article. We do not differ in our opinions about reliability, security and the like – and we concur that discovery, scalability, and accountability are also important to the future of Web services. But at Bloor, we think the other seven issues identified earlier in this chapter are the short-term must-be-solved issues that need to be addressed immediately in order to build enterprise-robust Web services architecture. Discovery, scalability, and accountability issues will be solved over time but are not, in our opinion, on the critical path at present for building robust Web services architecture.

Chapter Summary

At Bloor Research NA we have identified seven "shortcomings" of Web services architecture that may impede the progress of Web services in enterprise-class computing environments. In short, we believe the following to be Web services shortcomings that must be overcome in order to foster the adoption of the architecture in message-rich, transaction-heavy, enterprise-class computing environments:

- *Security/privacy* – specifically ensuring that content can be authenticated, and validated; can restrict authorization rights to prevent unauthorized viewers from using restricted content; and that ways be found to ensure data integrity.

- *Routing/messaging, Reliability/quality-of-service/transaction handling* – protocols must be developed to track messages through multiple intermediaries; and standards must be established to ensure messaging reliability (this is especially important in financial transaction processing scenarios where the underlying Web services architecture must be capable of "rolling-back" transactions to their original state should a failure of some sort occur. Until these protocol are developed (or become more mature), Bloor Research NA will continue to identify these areas as shortcomings.

- *Transaction-handling in particular* – all predecessor distributed computing architectures provided ways to monitor and track transactions, roll-back transactions, and manage/tune application and systems environments in order to ensure optimal performance. Many of these architectures made use of transaction monitors to accomplish this task. The W3C has not, and is not in the process, of creating a standard transaction monitor. Thus, this task will roll to vendors or open source software providers. And at present, the lack of a robust Web services standard for transaction tracking/roll-back/monitoring is, in the opinion of Bloor Research NA, a Web services shortcoming.

- *Manageability* – the lack of an overall plan for managing Web services environments (from a system, network, and application level) makes Bloor Research NA consider Web services manageability an architectural shortcoming. (Note that many vendors are stepping forward with robust, Web services-capable frameworks and products to assist in Web services management in the short term as well as for the long term).

- *Interoperability* – The formation of the WS-I goes a long way toward addressing Bloor Research NA's concern about how interoperability will be achieved across multiple disparate platforms and oft-times differing application program language environments. Still, the newness of the WS-I (no a long track record in resolving vendor-related issues) makes Bloor a little wary of declaring interoperability a fait accompli. Hence, interoperability is identified as a shortcoming.

- *Performance/tuning* – There is a distinct lack of performance tuning information and/or tools for optimizing Web services computing environments. Although the W3C does internal quality assurance (and has even published some of its test suites/tuning utilities), it is Bloor Research NA's opinion that not enough is being done here to assure that Web services applications can be optimized to compete from a performance perspective with tightly coupled distributed computing architectures. Hence, we see performance/tuning as an exposure to the general acceptance of Web services architecture by enterprise clients – and thus consider performance/tuning to be a shortcoming.

- *Device independence* – Simply stated, we see little aggressive activity at the W3C in terms of driving standards for multiple, disparate non-PC/non-server devices. And, to date most of the activity that has taken place has been centered on handling device graphics display/information preferences, some security, and tracking of standards activities by other standards setting bodies. Until we see more aggressive development of highly-functional non-PC/non-server implementations of devices, we will continue to list device independence as a shortcoming.

Although Bloor Research NA has identified seven areas that we consider to be shortcomings in Web services architecture, we note that a short article (found on IBM's DeveloperWorks web site) has identified three additional areas that should also be scrutinized. They are: discovery, scalability, and accountability. Although Bloor Research NA strongly agrees with the author's assertion that these areas need to be better fleshed-out, we do not consider them to be vital, must-be-addressed-immediately shortcomings.

All-in-all, Web services at present is a good messaging architecture for sending XML data using SOAP protocol. Also worthy of note is that the W3C has done a good job of standardizing the XML security/privacy protocols – so data/content can now be secured at the content level (as opposed to just securing data/content at the network/line level using encryption technology).

But for Web services to be accepted at the enterprise-level, this architecture must address several shortcomings – the most acute of which is how Web services handle complex transactions. At present, a piece of the ownership for addressing complex transaction handling belongs to the W3C (specifically, the piece that has to do with how messaging/routing protocols handle sending messages to single or multiple recipients and tracking the movement of those messages through intermediaries). The other big piece of the complex transaction-handling problem appears to belong to consortia like OASIS (which is developing its own business transaction protocol (BTP) to handle transaction roll-back and other tasks). Further, there are efforts underway at vendors like IBM, HP, Microsoft and BEA to create transaction monitor products that allow for tuning and better management of complex transactions.

The bottom line is that for Web services to be accepted at the enterprise-level issues such as reliability, quality-of-service, transaction handling, security, manageability, and more must be addressed in a cohesive fashion. And the good news is that, through the efforts of the W3C, the vendor community, open source software providers, and consortia – all of these shortcomings are being addressed.

What's Next?

This chapter described various shortcomings in Web services architecture that may prevent Web services applications from being used in mission-critical, production computing environments. It described these shortcomings and their impact on mission-critical application deployment in general terms (it identifies the problem and discusses the impact). Part II of this report (the next major section) takes a closer look at each of these shortcomings in depth – describing what the shortcoming is; what the standards committees are doing to rectify the shortcoming; and what vendors are doing to correct the situation until the formal standard recommendations are made.

PART II: Web Services State-of-the-Art

What This Part of the Report Is About

This section (Part II) focuses on examining Web services architectural shortcomings (security, reliability, transaction-handling, manageability, Interoperability, and so on) in greater depth. Each chapter considers the following points:

- *The Goal* – (for instance, "transactional security' or "routing/-messaging reliability");
- *The Issue(s)* – (issues that must be addressed to make the identified Web service enterprise-ready);
- *Standards Activity* – (a description of where Web services standards organizations have focused their efforts in order to remedy architectural shortcomings);
- *Vendor Remedies* – (a description of how certain vendors are addressing identified architectural shortcomings).

Following the research and analysis presented in this section, Part III seeks to describe what vendors, consortia, and open source software makers are doing to fill-in-the-gaps in Web services architecture until standards are developed and deployed in the general marketplace.

What We Hope You'll Learn

When all is said-and-done, we hope that you come away from this part of the report with an understanding of what needs to be done to address the shortcomings in Web services architecture. And we hope that you gain an understanding of where Web services solutions that are not part of the standards definition can be found that will help your organization fill-in-the-gaps using third party products to build reliable, secure Web services computing environments.

Chapter
4

Security/Privacy and Web Services

The Goals

Bloor Research NA believes that there are four aspects to consider when evaluating Web services security. They are:

1. Security over the network (line level encryption);
2. Security of documents and data (XML document/data security);
3. Personal information security (privacy); and,
4. Security best practices (policies and procedures).

The goal for potential adopters of Web services architecture is to address each of these considerations as your organization seeks to deploy Web services architecture.

The Issues

Let's examine each of these elements individually:

Security Over the Network

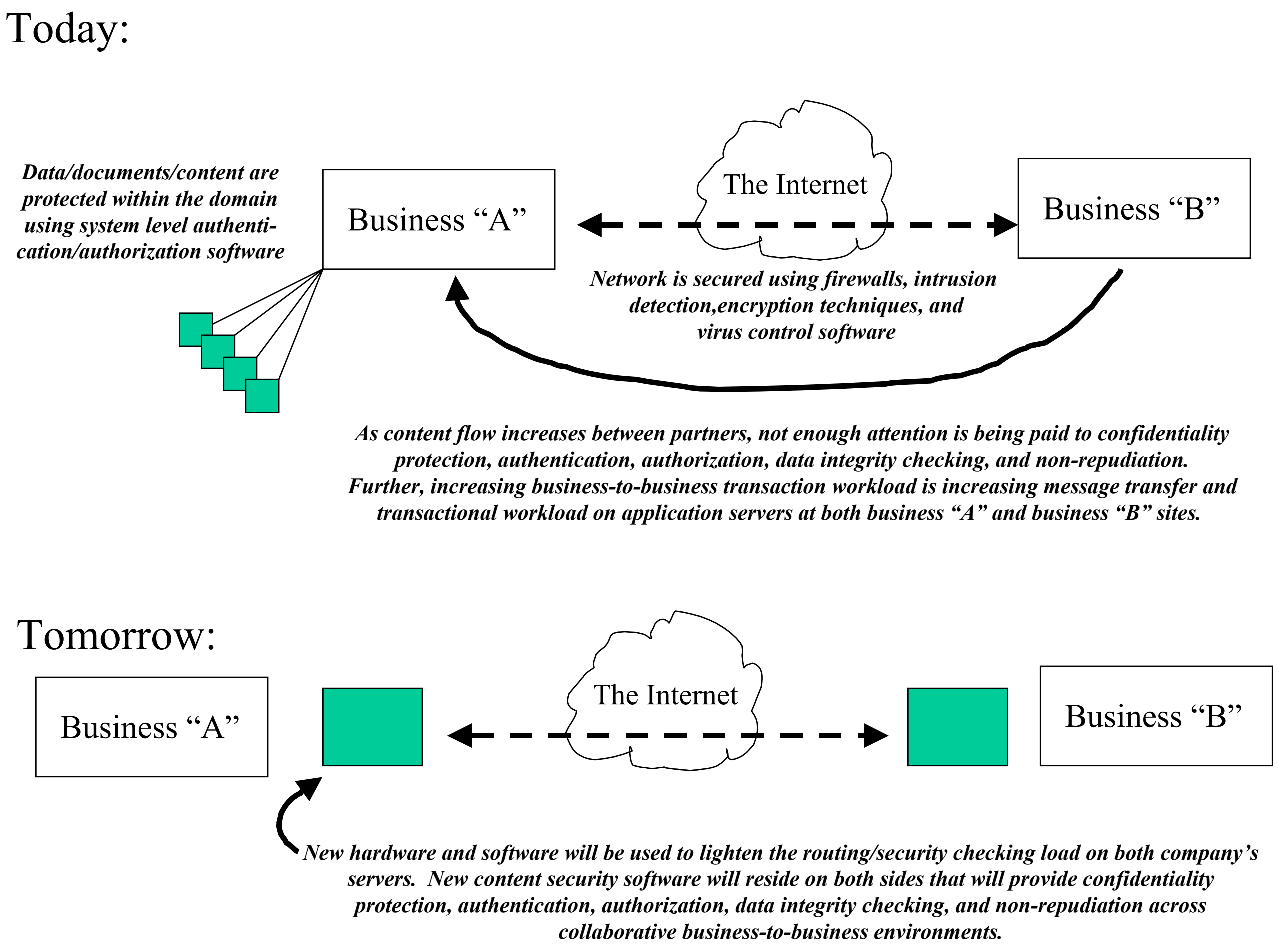
Designing networks that provide line level security (for information that flows over the Internet or across private intra- or extranets) is well understood by most organizations today. Information is frequently encrypted using public or private keys, and secured using the Secure Sockets Layer (SSL) protocol. If your organization is not familiar with how to do this there are numerous books available on this subject from many computer trade publishers.

Still, SSL is not the only issue to be concerned with when building a Web services architecture. It must be noted that the HTTP transport (the basic transport of the Internet) can tunnel through existing network firewalls and, using SOAP protocols, establish communications with applications within an enterprise infrastructure. If this scenario is not desirable, other steps need to be taken to pass Web services data securely. One approach to overcome this exposure is to use HTTPS (the secure, encrypted version of HTTP) coupled with Web services digest authentication standards. Another approach is to use richer middleware that can enable programs to be connected and that has built-in security features (this approach is discussed later-on in the Vendors/ Open Source/Consortia section of this chapter).

Security of Documents/Data
To Bloor Research NA, this particular area (document/data security/integrity) is where we believe the W3C has made the most progress in standards setting (a list and summary of W3C security standard recommendations is contained in the "Standards Activity" section of this chapter).

The issue at hand is to protect data and content while it is being sent and while it is stored (see Figure 10).

Figure 10 – Protecting Data and Content Between Business Partners


Today:
Data/documents/content are protected within the domain using system level authentication/authorization software
Business "A"
The Internet
Business "B"
Network is secured using firewalls, intrusion detection,encryption techniques, and virus control software
As content flow increases between partners, not enough attention is being paid to confidentiality protection, authentication, authorization, data integrity checking, and non-repudiation. Further, increasing business-to-business transaction workload is increasing message transfer and transactional workload on application servers at both business "A" and business "B" sites.
Tomorrow:
Business "A"
The Internet
Business "B"
New hardware and software will be used to lighten the routing/security checking load on both company's servers. New content security software will reside on both sides that will provide confidentiality protection, authentication, authorization, data integrity checking, and non-repudiation across collaborative business-to-business environments.

Source: Bloor Research North America, May, 2002

More specifically, security precautions need to be put in place to:

- Protect the integrity of data (*data integrity*) that is sent and stored;

- Ensure that confidential information is kept confidential (using an *authorization* list or policy to ensure that only individuals who are authorized to access information may do so);

- Ensure that data and documents received are from the source claimed (*authentication*); and,

- Provide an "audit trail" such that claims that deny that a service has taken place or has been provided can be refuted or confirmed (*non-repudiation*).

Various XML protocols now enable this type of security to be implemented using Web services and XML architecture.

Privacy

Another area in which the W3C has shown great progress is in the area of personal privacy protection. For years the computing industry has struggled with finding ways to ensure that personal private data be kept private. To address this issue, the W3C has initiated its Platform for Privacy Preferences Project (P3P), an initiative designed to create an industry standard for providing "a simple, automated way for users to gain more control over the use of personal information on Web sites they visit".

To this end, the W3C has enabled users to set their personal privacy preferences in machine-readable format that can be read by browsers. The browsers can compare the information that they desire to collect versus the information that the user is willing to supply – and act as instructed by the personal privacy program and the browser program. This P3P process, thus, gives users greater control over the use of their personal information on the Internet.

Web Services Security Best Practices

Most Information Systems executives know that security is not only comprised of standards and products, but also best practice policies and procedures for managing a computing environment. On the "best practices" standardization issue the W3C does not set in place best practice recommendations for securing Web environments. But, it does provide general advice that is useful for setting-up security policies and procedures, as shown below:

If you are a Webmaster, system administrator, or are otherwise involved with the administration of a network, the single most important step you can take to increase your site's security is to create a written security policy. This security policy should succinctly lay out your organization's policies with regard to:

- who is allowed to use the system

- when they are allowed to use it

- what they are allowed to do (different groups may be granted different levels of access)

- procedures for granting access to the system
- procedures for revoking access (e.g. when an employee leaves)
- what constitutes acceptable use of the system
- remote and local login methods
- system monitoring procedures
- protocols for responding to suspected security breaches

This policy need not be anything fancy. It need only be a succinct summary of how the information system work, reflecting your organization's technological and political realities. There are several benefits to having a written security policy:

You yourself will understand what is and is not permitted on the system. If you don't have a clear picture of what is permitted, you can never be sure when a violation has occurred.

Others in your organization will understand what the security policy is. The written policy raises the level of security consciousness, and provides a focal point for discussion.

The security policy serves as a requirements document against which technical solutions can be judged. This helps guard against the "buy first, ask questions later" syndrome.

The policy may help bolster your legal case should you ever need to prosecute for a security violation.

Source: W3C Web site at http://www.w3.org/Security/Faq/wwwsf1.html#GEN-Q7

Standards Activity

XML-based Security; Privacy; and SOAP Security Standards Efforts

Although XML is not the focus of this report, it is important to note that much of the W3C standards activity that has taken place has been based upon providing authorization, authentication, confidentiality, data integrity, and non-repudiation services for Web services users.

A close look at the W3C standards committees involved in setting security recommendations show that these committees have focused primarily on making product-oriented recommendations in three XML-related areas, one privacy area, and two network communications areas (SOAP and HTTP). The W3C is currently not in the process of establishing “best-practice” policy and procedure recommendations (although it does include useful “advice” for setting-up security policies and procedures as part of its Security FAQ (frequently asked questions).

The key areas of focus by the W3C in the area of security are:

- *XML Encryption* – W3C standards committees are working to develop a process for encrypting/decrypting digital content (including XML documents). It is also responsible for creating an XML syntax used to represent encrypted content and information that enables an intended recipient to decrypt it. This working group is important to track because it plays a vital role in ensuring the security of documents and data sent and received over an underlying Web services architecture.

- *XML Digital Signature* – W3C standards committees are working to develop an XML compliant syntax used for representing the signature of Web resources and portions of protocol messages – and for putting in place the procedures for verifying those signatures.

- *XML Key Management* – W3C standards committees are working to develop XML protocols that allow a client to obtain key information (values, certificates, management or trust data) from a Web service. This group is important because it creates important standards for sending and receiving information and data to/from trusted sources.

- *The Platform for Privacy Preferences (P3P)* – W3C standards committees are working to provide "a simple, automated way for users to gain more control over the use of personal information on Web sites they visit". It is important to follow because it helps set standards that users will eventually use to control the type of private information they put forward on the Web.

- *SOAP* – W3C standards committees have already specified how XML signatures should be handled by SOAP (via standardized envelope headers). And SOAP-SEC (SOAP security) already has a mechanism for handling encrypted messages. More work, however needs to occur to support multiple protocols (protocols with richer security features); as well as additional work needs to be performed to strengthen overall SOAP over HTTP security.

- *HTTP* – W3C standards committees have already specified a scheme for authenticating the identities of users. This scheme is known as "digest authentication". HTTP is great as a low-level protocol for simple messaging – but Bloor Research NA believes that other, richer protocols such as UDP and TCP/IP will eventually be used to improve security and reliability (HTTP is probably just too simple to provide the kind of secure and reliable network connection desired by most enterprise application architects).

The most important W3C activities to watch in the security space include:

- XML Encryption Activity Group – The XML Encryption Activity specifies XML encryption syntax and processing for encrypting XML.

- XML Key Management Activity Group – The XML Signature and XML Encryption Activities focus on signature and encryption. In March, 2002, the XKMS (Key Management) Working Group published its XML Key Management Specification (XKMS 2.0) that specifies protocols for distributing and registering public keys for use with XML Signature and XML Encryption. When this specification becomes an official recommendation the standards protocols needed for distributing and registering public keys will be in place.

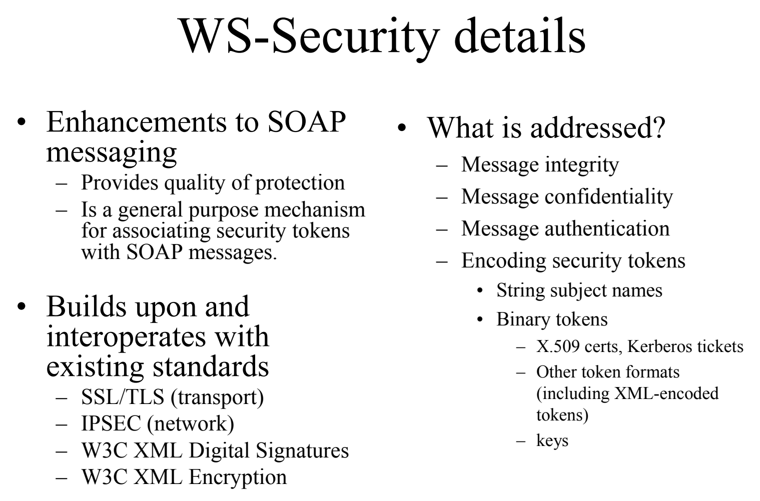
- XML Key Management (2.0) Requirements, in Last Call through 15 April, specifies design principles and scope. X-BULK allows bulk registration necessary for systems such as smart card management. Comments are welcome. Visit the XKMS home

- The XML Signature Activity Group – This activity focuses on providing integrity, signature assurance and non-repudiatability for digital signatures that are sent over the Internet. Digital signatures are important for use in conducting electronic commerce (for instance, digitally signing a credit card purchase) or for other legal or contractual reasons.

- P3P – The Platform for Privacy Preferences Project released its 1.0 recommendation in January of 2002. Ongoing activities will include implementing XML signatures and authentication as well as dealing with privacy protection on wireless devices.

Independent Activities
On April 11, 2002, IBM, Microsoft, and Verisign announced a suite of specifications called *WS-Security* that extend SOAP security and build on other existing Web services security standards. These "extensions" (which are being proposed to OASIS) are described in Figure 11 (below).

Figure 11 – Vendor-proposed Security Extensions



WS-Security details

- Enhancements to SOAP messaging
  - Provides quality of protection
  - Is a general purpose mechanism for associating security tokens with SOAP messages.
- Builds upon and interoperates with existing standards
  - SSL/TLS (transport)
  - IPSEC (network)
  - W3C XML Digital Signatures
  - W3C XML Encryption
- What is addressed?
  - Message integrity
  - Message confidentiality
  - Message authentication
  - Encoding security tokens
    - String subject names
    - Binary tokens
      - X.509 certs, Kerberos tickets
      - Other token formats (including XML-encoded tokens)
      - keys

Source: IBM: Security in a Web services World: April 10, 2002 Analyst Briefing

These specifications are particularly noteworthy for two reasons: 1) they do directly address several shortcomings related to Web services security; and 2) they are vendor "recommendations" (not formally approved W3C standards).

Point number 2 is worth dwelling on. Sometimes when standards committees take too long to formulate and approve official standards, the vendor community steps-in with workarounds that provide functionality needed immediately by the market until formal standards are approved. Such is the case with WS-Security. WS-Security fills known holes in Web services security and helps make it possible to use Web services architecture in production, mission-critical environments. Over time, we at Bloor Research NA expect WS-Security extensions to become formal standards (via OASIS).

Many OASIS member organizations plan to support for WS-Security, including Baltimore Technologies plc., BEA Systems Inc., Documentum Inc., Entrust Inc., IONA, Netegrity Inc., Novell Inc., Oblix Inc., RSA Security Inc., SAP AG, Sun Microsystems Inc. and Systinet Corp.

Vendor/Open Source/Consortia Remedies

In short:

- Information Systems managers already understand the basic concepts and products that can and should be used to secure a network. Firewalls, encryption, denial-of-service (and other types of attacks) and SSL have become standard fare in the lexicon of systems/network administrators. (If you need more information on these concepts there are dozens upon dozens of books that can familiarize you with the subject). Hundreds of products can be purchased from network vendors to assist in building secure networks.

- It is important to note that Web services SOAP protocol over the HTTP transport enables certain applications to tunnel through existing firewalls – thus creating an opportunity for a security breach. IS managers concerned with using SOAP/HTTP should consider using other protocols such as HTTPS, or should consider using other middleware for connecting application programs (such as IBM's MQSeries or Microsoft's Windows Message Queuing) because these middleware transports have built-in security features.

- To further augment Web services security steps should be taken to ensure content security. To provide security for documents and data that have been sent over the Internet, XML standards have been specified to address non-network security considerations such as authentication, authorization, data integrity, and non-repudiation. These standards are working their way into vendor products from Microsoft, IBM, HP, Sun, BEA, IONA, webMethods, and more (the security offerings of many of these vendors are listed in Chapter 9 of this report). Open source products such as the W3C's Jigsaw application server also has XML security standards implemented as part of the source code.

- With respect to privacy considerations, IS buyers as well as consumers should become familiar with Microsoft's Passport and The Liberty Alliance personal identification/authentication Web services. The basic concept of these two initiatives is to create a single system log-on identity that would enable a user to identify himself/herself to a variety of different programs on the Internet and to provide personal data to those programs. These Web services help overcome the problem of constantly having to fill-out name, address, city, state/province, country information by providing a single identification "profile" from which this information can be gleaned. This is Web services privacy in action – the thing to watch-out for is that personal information that is captured by Passport or the Liberty Alliance is not exposed through security flaws or hacker efforts.

Chapter Summary

In our opinion, most of the current W3C standards activity is taking place on XML. XML Encryption, XML Schema, XML Signature, and XML Protocols are the key foci – but additional work is being done on XML and its relationship to DOM, RDF, CSS, XSL, XHTML· MathML, SMIL, SVG, and Canonicalization. (*Note: XML is not the focus of this report* – the shortcomings in Web services architecture is. *But it is worthy noting how much effort is being put into strengthening XML* because ultimately it is the XML data that is read by cooperative applications – Web services protocols and the registry service are just ways that applications find each other and communicate with each other).

Bloor Research NA observes that for most enterprises, computer security consists of both establishing best-practice policy and procedures as well as incorporating the use of security software. We note that although the W3C has already made progress in creating security recommendations at the protocol and content levels, there is little work going on at present to address security management and security best practices. (We do acknowledge that the W3C does provide security best practices advice as part of its Security FAQ – frequently asked questions – section. Also, we believe that the W3C needs to keep its security priorities very focused in scope – and not try to be all-things-to-all-people. We therefore agree with the areas of focus that the W3C has chosen in the security field).

Having stated our agreement in principle with the W3Cs standardization focus, we note that the vendor community has aggressively stepped forward to fill-in-the-gaps in Web services security – particularly at the network and application management levels.

These security standardization efforts are important because they mean that prospective Web services users can rely on the W3C for data/content security and for information privacy – but they will need to turn to vendor-based options to supplement Web services architecture as they wait for formal standards to mature. Vendors such as IBM, Verisign, and Microsoft are currently stepping-in to help fill-the-security-gaps in Web services architecture with recommendations such as WS-security that we at Bloor Research NA believe will ultimately become standards. Other vendors such as Actional, Netegrity, RSA, Stele, and Vordel have also come forward to offer specialized, Web-services enabled security products that can step-in and fill-the-gap in security at the network level (as well as potentially in security management). In short, Web services security has reached the point that it can be implemented and operated in enterprise-class, mission-critical computing environments.

Chapter
5

Routing/Messaging; Reliability/Quality-of-Service; and Transaction Handling – They're All Intertwined

The Goals

Simply stated: Web services architecture is a message-passing architecture. The passing of services-related messages and data/content between applications is core and fundamental to Web services architectural design. It is therefore paramount that messaging be conducted expeditiously and reliably. And, it is also important that messages be capable of being traced/tracked from sender, through intermediaries, to recipients.

If messages cannot be sent reliably, businesses will not use Web services architecture for transaction processing. If messages take too long to reach their destination or become bogged-down in a processing queue once they have reached their destination, Web services applications will not meet performance acceptability expectations set by users. And if messages cannot be traced/tracked, then responding to senders and intermediaries will be impossible – and troubleshooting will be difficult. The goals, therefore, of Web services from a routing/messaging perspective are:

1. Perform reliably;
2. Perform expeditiously; and,
3. Ensure that there is an audit trail.

The Issues

Routing/messaging, reliability/quality-of-service, and transaction handling are all intertwined. For transactions to be processed successfully they have to be routed to the right place and processed. And features such as transaction roll-back (a process that kicks-in should an application fail whereby the transaction is stepped back to its original state) need to be in place to ensure that transactions can be conducted reliably. If either of these characteristics/features is not in place, transactions will fail, data will be corrupted, and there is little to nothing that a systems administrator could do to repair the situation.

Further, Web services reliability cannot and should not be solely confined to whether the application processes remain “communicative” – reliability also depends on systems and network variables that could affect the availability of a given Web service. Should a system fail, all sort of questions arise such as “where can a like service be obtained”; or if the system that fails has the data that needs to be processed “is there an alternative way to get to the data”. In short, reliability does not only imply Web services program-to-program reliability, it also implies that in some cases systems be either highly-available or fault tolerant.

From a routing/messaging perspective, for Web services architecture to succeed in business computing environments it needs three characteristics: reliability, acceptable performance, and traceability.

- By “reliability” we mean that messages must reach their respective destinations (and failing to do so, must be capable of rolling-back to the original message/transactional state);

- By “manifest acceptable performance” we mean that applications cannot get “hung-up” waiting for messages to be processed. Cooperative applications must appear to the user as one cohesive, seamless application environment that meets end user performance requirements; and,

- By “traceability” we mean that messages need to be able to be tracked between sender, receiver and any intermediaries involved. This information can be used for troubleshooting or as a means to correlate a response.

Message Handling More Acute With Web services

It is also important to point out that Web services architecture is more dependent on fast, accurate messaging than predecessor distributed computing architectures. Our justification for this statement is that Web services architecture enables applications to act independently – applications can automatically go to the Internet and search for services, negotiate how services are to be provided, and then invoke services to begin. In other words, Web services programs ask a lot more questions (send a lot more messages) than other distributed computing architectures (these other distributed computing architectures are told where to go to get services because a service requester application is hardwired to a service responder application). Applications in predecessor architectures know where their cooperative partners are – and how to interoperate with those partners – and accordingly do not need to send as much message traffic as called for by Web services architecture.

A closer look at the Web services message-passing scenario shows why Web services generate a large increase in message traffic. As Web services architecture matures, applications will request UDDI services in order to locate cooperative applications. Once those cooperative applications have been located, negotiations between applications regarding how to work together using WSDL begin. Then, actual data or content is passed over the network using SOAP and HTTP to bind and link messages. Other distributed computing architectures do not necessarily require all of these services (for instance, in a tightly coupled program-to-program scenario applications are hard-wired together so no UDDI registry look-up is required; and applications know how to communicate with each other so no negotiation and related message passing has to take place in order to make cooperative applications work together).

It is fair to say that one major consideration when build a Web services architecture is: *plan for your systems to handle increased messaging traffic*.

Standards Activity

The basic W3C bottom line with routing/messaging is that the W3C takes direct responsibility for creating the infrastructure and protocols needed for applications to send and receive data (as well as keep records of intermediaries). The W3C is not involved in setting standards for how to provide for transaction roll-back; nor is the W3C taking an active role in creating a transaction monitor to help manage and/or tune transactions.

So, at the infrastructure/protocol level, the organization to track at the W3C in order to follow routing/messaging standardization is the XML Protocol Working Group (the group that helps devise SOAP standards).

The W3C Ecommerce/Micropayment Activity Is Now Closed!
When searching for information regarding Web services transaction processing on the W3C web site, researchers eventually come to the following message:

"The W3C Ecommerce/Micropayment Activity is now closed".

Does this mean that the W3C has already figured out how to standardize transaction processing and payment collection for services rendered? Far from it!

The W3C Electronic Commerce Activity did generate specifications for handling micropayments. (The specification for handling micropayment per-fee-links was published as a W3C final Working Draft on 25 August 1999). This specification allows merchants to propose multiple payment systems to prospective buyers. And it allows users/buyers to have more than one method of payment in operation from their browser. Hence, the micropayment activity has been closed because the specification to handle micropayments already exists. But the need to help provide standards for conducting complex transactions lives on.

Vendor/Open Source/Consortia Remedies

The W3C knows that there is a need to provide a basic infrastructure for conducting reliable and secure transactions over the Internet using Web services architecture. They have also identified the need for "low friction" commerce transactions that make it easy and transparent for users/customers to transact business over the Internet. But as the W3C puts it: "the role of W3C is to focus on core infrastructure technologies for Electronic Commerce and identify common infrastructure needed in this area. W3C is not committed for example in specifying banking systems nor schemas for specific Electronic Commerce applications".

To the end of providing the infrastructure needed to conduct eCommerce, the W3C has specified protocols for XML digital signature verification; for XML encryption (for sending secure data and content); and even the XML protocol itself allows two or more peers to communicate, making it possible for applications to communicate and negotiate. But these standards primarily address "lightweight" transactions – to support more complex transactional environments an industrial-strength Web services transaction handling scheme or architecture needs to be put in place.

And such architectural recommendations are now starting to come forward from various consortia as well as vendors:

- One of the more interesting approaches to complex transaction processing is being proposed by the OASIS consortium and it is known as the Business Transaction Protocol (BTP). And IBM, Microsoft, HP and others have their own approaches to complex transaction processing under development.

- Other vendors are approaching the problem of reliability from a communications/networking perspective. Kenemea offers a message switch that offloads systems from having to do the bulk of Web services message processing while at the same time improving overall systems reliability. Blue Titan and Flamenco Networks build network meshed environments that provide for increased reliability as well as security and management.

- To remedy potential systems/network failures that may occur when processing complex transactions, various vendors are starting to introduce Web services environments that are highly reliable or even fault tolerant (they won't fail). One such vendor is Zebrazone – a company that promises "eternal availability" with its uptime management platform. This platform is a massively parallel clustered systems architecture.

Point Products Are Becoming Available

Also worthy of note is that several vendors have stepped forward with point products to help address billing and payment for Web services transaction services provided. MetraTech currently serves as a great example of such products – the company offers a Web services enabled billing, revenue sharing, and settlement package that makes it possible to structure a pay-for-service platform between sellers and buyers.

Chapter Summary

One of the most fundamental concepts involving distributed computing architecture is "*store-and-forward message passing*". The way that distributed computing architecture actually works is based-upon constantly sending messages to other applications that perform services for the original requester application.

But handling increased message processing load is not the only messaging/routing-related issue to be dealt with. Messages need to be able to be passed reliably between programs – and should a program fail, messages (in particular transaction messages) need to be rolled-back to their original state. To this end the W3C is working on protocol standards for reliable messaging between sending and receiving applications as well as tracking messaging through intermediaries and correlate replies to all involved parties.

As the W3C continues to work on transaction processing and messaging from a protocol and infrastructure perspective, the vendor/consortia communities are looking to solve complex issues related to ensuring data integrity, transactional-roll back, and highly reliable messaging. Many vendors have their own approaches to handling complex transaction processing (some of these vendors approaches are listed in Chapter 9). And some consortia have also articulated their view of how complex transaction processing should take place (most notably the OASIS consortium has articulated its Business Transaction Protocol). Finally, point products are starting to arrive to address specific transactional issues such as micropayments.

From Bloor Research NA's perspective, it's hard to imagine that any "standard" way of building a Web services-based complex transaction processing architecture will come to exist in the near future. There are just too many variables such as systems architecture (is it fault tolerant and does it need to be; or network bandwidth; or business practices and processes) that influence how reliably transactions can be processed. And how applications are constructed and where they seek services, and which types of services they seek will also influence the reliability of messaging and of transaction handling. For these reasons, we conclude that the W3C is focused in the right place – at the infrastructure level establishing messaging protocols that manage and track messages between senders, receivers, and intermediaries. These protocols will ensure that any complex transaction-handling systems will have a common basis for communications using W3C message standards.

The delightful aspect of messaging/transaction handling is the way that vendors and consortia are stepping forward with solutions that augment the W3C protocols. Some of these solutions were described in this chapter (fault tolerant architecture, BTP, microtransaction point products, etc.). But the good news for buyers of Web services is that point products, full-fledged vendor solutions, and consortia recommendations already exist or are forthcoming – all augmenting Web services shortcomings and making it possible to use Web services in mission-critical computing environments today.

Chapter
6

Manageabilty

The Goal

Because Web services are heavily based on program-to-program communications, the ability to manage applications is vital to the operation of a Web-services-based systems and network environment. Systems administrators need management consoles, tools, and utilities that can provide them with the status of:

- The Network;
- Systems; and
- Applications.

It is in the area of applications management that Web services architecture requires extra attention. Not only do systems administrators need to be able to understand what is happening when an application request for service breaks down, or why performance may be slow – they also need to understand how to manage and *orchestrate* a myriad of application objects that work together to create one composite application.

> Businesses and people that work together need their applications and services to work together. That's why the industry is moving towards Web Services.
>
> Making Web Services work is a two step process. First you publish and then you orchestrate. Publishing means making the services available, orchestrating means coordinating asynchronous services into manageable business applications.
>
> You can think of orchestration as the logic and rules that assemble multiple synchronous and asynchronous web services into a long-lasting multi-step business transaction/process.
>
> Executing, monitoring and managing orchestration logic is complex. This is why a new class of software infrastructure called Web Service Orchestration Server is emerging.

Source: http://www.collaxa.com/orchestration.jsp

Should a problem arise, is the network at fault? Is the service being requested available? Is there a performance issue processing the request? Web services architecture places additional demand on administrators to understand what is happening at a messaging level between application objects within a computing infrastructure at a very granular level.

But in many respects, understanding where message passing is breaking down – and then fixing broken routers or restarting servers or applications is “old think”. “Management of the business process” is the latest in-vogue concept related to systems/network/applications management. The concept is founded on the principle that it’s the execution of the business application process (for instance, a purchasing transaction) that’s important – downed routers, systems overload, and other elements are just contributing causes of the problem. (Systems administrators need to consider that fixing a downed router may alleviate a problem, but it might not be the most sensitive aspect of fixing a broken transaction process). Process management emphasizes taking a holistic view of a breakdown and fixing the problem accordingly, rather than taking a stab in the dark at fixing broken components in a piecemeal fashion.

So, the goals in management of Web services architecture should be to provide:

1. Traditional management of applications, servers, and the network (augmented with additional monitor/control for granular application objects); and

2. Management of the overall business process.

The Issues

Traditional systems/network/application management products can be used to manage Web services architecture and related information infrastructure. Tons of tools, utilities, and software programs exist today that give systems administrators a good view of networked distributed systems environments. Products from Computer Associates, Hewlett-Packard, IBM, Microsoft and a dozens of other vendors provide for a centralized view of a traditional distributed computing environment – and make it possible to monitor, control, tune, and administrate such environments.

Still, Web services architecture changes the emphasis of these management products a bit. It calls for the management of application objects at a granular level – and for business processes at a very high conceptual level. And many traditional management products are not capable of providing the level of drill-down needed to manage and orchestrate a myriad of application objects across multiple, disparate platforms.

Further, one of the biggest “paybacks” of using Web services architecture will be that it allows enterprises to build applications in such a way that business process flow can be easily streamlined. And streamlining business process flow can result in time saving for sales, general, and administrative personnel (savings that can be passed directly to an enterprise’s bottom financial line). So Web services management products should incorporate the ability to link applications to processes – and provide administrators with a view of such processes – in order to result in maximum payback due to application/process streamlining.

In short, a new generation of application tools and utilities need to be developed that allows for the monitoring and control of application objects; while also allowing applications to be viewed as contributors to overall business process flow.

Standards Activity

The W3C does not focus on systems, network or applications management. Again, the W3C primarily focuses on infrastructure-related standards (such as communications protocols and middleware programmatic interfaces). Creating standards for management products is outside the scope-of-focus of the W3C.

Having said this, the W3C knows that many of its recommendations must have the ability to report activities to some management source application. So, where appropriate, the W3C is working toward putting such management reporting information into the WSDL such that it can be relayed to programs provided by vendors to allow for granular object/application management.

Vendor/Open Source/Consortia Remedies

Almost all of the systems/network/application management activity comes from the vendor community (some performance management software and some basic systems/network management software is available from open source sources).

The vendor offerings can be separated into several classes:

- Traditional system/network/application management software packages that provided monitoring, control, performance, report generation facilities, and other management functions;

- A new class of management appliance that makes use of specialized Web services SOAP "switches" that provide a central hub for message switching as well as a centralized point for managing Web services message passing. These specialized "appliances" are becoming available from companies such as Forum Systems, Blue Titan, Kenamea, and a host of others – and provide a central message-passing location that contains all of the information needed to track Web services messaging activities. (Further, these SOAP switches often provide additional security features (authentication, authorization, data integrity checking, and so on).

- Point products.

Traditional systems/network/application management vendors such as IBM, Microsoft, Hewlett-Packard, and Computer Associates are have products that can manage the various aspects of a distributed computing architecture. But, all of the vendors are introducing management software that can be used to examine problems at a business process level. And by so doing, enterprises will be better able to see the effects of downed hardware or services – and take the most appropriate action that will have the biggest effect in order to correct computing-related problems.

New vendors are making the scene with products designed to provide a granular view of application objects such that systems administrators can better troubleshoot problems or tune applications for better performance. Some of these vendors include Collaxa, Grand Central, Infravio, and Blue Titan to name a few. And not only do these new products provide a granular view of underlying application objects – several also provide the software needed to orchestrate Web services (ensuring that all components work together toward enabling a larger composite application or business process to complete its work).

Actional, Altoweb, and West Global are all examples of management point products that can be used to provide specific management functions for Web services applications. For instance Altoweb provides deployment, testing, and monitoring services for Web services applications.

Chapter Summary

Web services management involves:

1. Monitor and control of a myriad of application messages and objects;

2. Orchestration of applications that are formed by loosely-coupling objects into composite applications and business processes;

3. Viewing application processing not only from a granular what's-going-on-at-the-object-level – but also from a what's-going-on-from-a business-process-perspective.

The W3C does not concern itself at present with setting application management standards – instead focusing on protocol and infrastructure standardization. The W3C does recognize, however, that it must provide a way for applications to communicate information that is of use to third party applications management software – and will probably make room for such information to be presented as part of WSDL messaging to third party application management consoles.

From an overall systems/network/applications management perspective there are currently many software packages available from multiple vendors that provide for centralized management of distributed computing environments. On the other hand, Web services architecture demands that management software be able to determine application object behavior at a very granular level – and then provide a means to take any corrective action needed to correct any malfunctions that occur. And this demand for greater granularity calls for a new generation of Web services application management software.

Further, the fact that large, composite applications and business processes can be formed by linking dozens of small application objects together necessitates the need for application "orchestration". And again, a new generation of Web services application management software is called for to accomplish this orchestration.

Fortunately, the vendor community has stepped forward with software that can provide the level of analysis needed to ascertain what is happening at the atomic application object level, as well as in the bigger-picture orchestration level. Chapter 9 contains names and descriptions of several vendors that provide such products.

One last point to consider is that a new trend in systems/network/application management is to view a malfunctioning application from a business process perspective. Instead of just focusing on fixing a broken router, or restarting an application, this new trend emphasizes examining how a broken component is affecting the overall business process. And new systems/network/application management software is being developed that helps enterprises make more informed and effective decisions on how to fix problems based upon how the problems affect process flow. Given that Web services and business process management are closely related, IS buyers should keep a close eye on how these new process-oriented management applications can be applied to better manage Web services environments.

Chapter 7

Performance Tuning and Optimization

The Goal

"Beauty" is not the only thing that is in the eye of the beholder. "Acceptable" performance" is also in the eye of the beholder. And Web services applications have the potential to be performance disaster areas if applications are not designed, optimized, and tuned properly.

The best example that demonstrates the need for performance optimization is a simple transaction. Suppose that a prospective customer wants to order a part from your company. And suppose that the part requested is out of stock. A web service application could then be invoked to request your suppliers to provide you with a particular part within a particular time frame. Now, just suppose that this Web services application is slow to check your existing inventory – and even slower checking with your supply chain partners to find the part. How long do you think your prospective customer will be willing to wait until he or she cancels the transaction and seeks the part from some other organization?

It is important that Web services provide timely services – services that meet the requirements of the user in a timeframe that is acceptable to the user. The goal, therefore, in performance is to ensure that Web services applications can live within the allotted performance metrics of the users of those applications.

The Issues

Many Web services applications rely heavily on making a bunch of loosely coupled application objects work together in unison to form one larger application. Accordingly, any glitch in any underlying application object could cause an entire application to crash – or at least stall for a while until another like service could be found to replace the downed object.

In the previous chapter we discussed how new "orchestration" software is coming to market to help manage the ebb-and-flow of Web services applications that are comprise of multiple, cooperative application modules that form a much larger application or business process. But orchestration software is only one piece of ensuring that Web services applications stay up-and-running and perform within user-acceptable performance ranges. Another element has to do with tuning systems, networks, and the applications themselves to reach desired performance levels.

To this end, loads of systems management software, tools, and utilities exist that make it possible to optimize servers and networks for optimal performance. But what kind of tools and utilities exist to do the same for Web services application environments?

The next two sections examine this question?

Standards Activity

The W3C's Web services mission concentrates on protocol and infrastructure standardization that enables loosely coupled applications to communicate and share data. The organization knows that acceptable performance is a key requirement to be met in order to enable Web services to be accepted in mission critical computing environments. But the organization also knows that many factors outside of its reach influence how applications perform – including how systems and networks have been designed and configured.

The W3C does some performance testing and platform optimization as it seeks to test its standard recommendations on its open source "Jigsaw" application server. And the W3C publishes the results and its comments on its own testing efforts on its Web site (http://www.w3.org/Jigsaw/User/Introduction/performance.html). But the W3C is not in the testing and performance optimization business – and hence no "standards" for performance optimization are scheduled to be published by the W3C in the near or long term.

Vendor/Open Source/Consortia Remedies

Like manageability in the previous chapter, performance optimization efforts largely fall under the umbrella of the vendor community. Vendors have, for decades, provided tools and utilities that enable IS administrators to tune and optimize their systems and networks – and more recently have focused on providing tools and utilities that allow for performance and optimization tuning for application environments.

In general, there are two sources for tools and utilities for tuning Web services environments. They are:

- *Manageability frameworks* – such as the software packages described in the preceding manageability chapter. Manageability software already monitors and collects information and statistics on application use and behavior. So, providing the tools and utilities needed to optimize the performance of various applications is a natural extension for many management software packages.

- *Point products* – there are some open source products that can be used to tune and optimize Web services applications. And, there are also several vendors who have developed software specifically designed to help Web services applications perform optimally.

From a "manageability framework" perspective, many of the vendors mentioned in the previous chapter also provide tools for optimizing Web services performance. Some of these vendors include: Altoweb, Collaxa, and West Global.

From a point product perspective open source code called "Push-to-Test" provides a good testing environment for performance and scalability trials. And West Global's tools and utilities may also be used as point products to address performance and tuning needs. Other products from application development environment software makers as well as from several of the big systems makers can also be used as point products to tune Web services systems, networks, and applications.

Chapter Summary

Orchestration management software helps ensure that various Web services application objects can work in a unified fashion to accomplish computing tasks. But the use of orchestration management software does not necessarily guarantee that applications have been tuned and optimized for performance. To this end, IS managers must seek out tools and utilities that lend themselves to optimizing networks, systems, and most of all applications performance.

There are two sources of supply for such performance tuning tools and utilities: 1) management frameworks (such as those discussed in the previous chapter); and 2) point product solutions.

With regard to management frameworks, several vendors provide tools and utilities designed specifically to help tune loosely coupled Web services applications. Altoweb, Collaxa, and West Global are but a few of the newer companies that do so. With regard to point products, such products are usually found in the vendor community from suppliers such as IBM, HP, Sun, Microsoft, et al – but several new products have also made it to market as of late. Also, some open source performance testing tools/environments also exist (see Push-to-Test write up in Chapter 9 for further details).

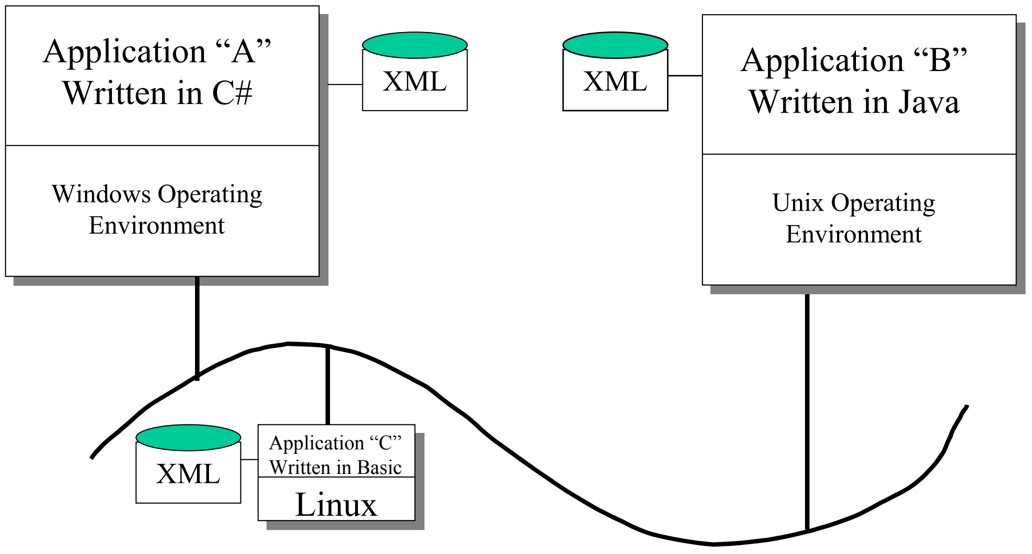
Chapter 8

Interoperability

The Goal

One of the goals of Web services is to enable applications that reside on diverse, disparate systems platforms to transparently interoperate with applications that reside on other systems architectures (irrespective of what programming language, operating environment, or database is being used on each platform). In other words, one of the most fundamental goals of Web services is "interoperability".

Figure 12 – Cross Platform, Program-to-Program Interoperability


Application "A" Written in C#
Windows Operating Environment
XML
XML
Application "B" Written in Java
Unix Operating Environment
XML
Application "C" Written in Basic
Linux

***XML** – A Common Format*

***SOAP** – Communications APIs/RPCs*

***WSDL** – Negotiate How to Work Together…*

***UDDI** – Used to Help Find Cooperative Applications*

Source: Bloor Research NA – May, 2002

Figure 12 shows three different operating environments and three different programming languages used on three different systems platforms. Using Web services each application can communicate and interoperate with other disparate applications by using XML to create a common format for sharing data/content, using SOAP for communications, using WSDL to negotiate how and what to communicate; and possibly using UDDI to locate application partners.

Cross-platform, program-to-program interoperability has been a goal for IS managers for almost thirty years. The reasons:

- If an IS manager is assured that his or her applications can communicate with each other regardless of which platform he or she chooses, then vendor lock-in is difficult to achieve. Instead of having to buy the same brand of processor with the same operating environment from the same vendor in order to achieve interoperability, IS managers would have more freedom of choice all systems had the ability to automatically interoperate.

- A system could be purchased on the basis of low cost, or good manageability, or using any of a number of other criteria because the IS buyer would know that, regardless of which system he or she purchased, it would automatically work with other vendor's equipment within the enterprise information infrastructure.

- Systems/application integration costs run some enterprises millions of dollars annually. These costs could be dramatically reduced should guaranteed interoperability between disparate systems architectures be achieved.

So, some of the reasons that interoperability is important to IS managers are:

- Flexibility – Increased flexibility (ability to choose any vendor's products and have them automatically work within an existing information infrastructure);

- Platform cost – No more proprietary vendor lock-in. IS managers gain new leverage in system purchase activities; and

- Reduced systems/application integration costs. (Millions of dollars are spent annually integrating disparate systems and applications. Suddenly, by using Web services protocols, registries, and XML format, these costs dissipate).

The Issues

So, if guaranteed interoperability is so desirable, why has it taken so long to achieve it? An examination of past attempts to achieve interoperability reveals several of the obstacles that have surfaced over the years that have prevented true system-to-system, application-to-application interoperability from being achieved.

Past Failures in Interoperability
In the past, predecessor distributed computing architectures have failed to win mass following in the distributed computing marketplace, or met with moderate success for three reasons:

1. They were considered proprietary (politics);
2. They were too difficult to implement;
3. Vendors implemented standards differently.

Web services architecture solves problem #1 in that it is not considered proprietary. Quite the contrary: with over five hundred vendor/end-user organizations signed-up to assist in Web services standards definition and development, it would be difficult to argue that Web services could be accused of being "proprietary".

Previous attempts at building an industry-wide, mutually acceptable distributed computing architecture often failed because the standards implementations were too difficult to implement. Witness the Open Systems Interconnect standards of X.400 and X.500 (messaging and directory standards). These standards were superior in depth and breadth to the defacto TCP/IP equivalents for message passing and directory services (Simple Mail Transfer Protocol – SMTP, and Yellow Pages domain name services). Yet the TCP/IP defacto standards won out and became the basis for Internet communications. Why? One major reason was that the OSI standards were difficult to implement. Web services, due to their simplicity-to-implement may help overcome problem #2.

But problem #3 is a bit more difficult to solve.

Why Interoperability Organizations Generally Fail
Another reason that standards like OSI fail to achieve grand market acceptance is because various vendors either partially implemented the standards (leading to incompatibilities with other vendor's fuller implementations). Or vendors used their own liberal interpretations of X.400 when they built products (for instance to incorporate an existing mail system with the standard), again leading to incompatibilities.

To overcome these types of implementation issues, standards setting organizations or independent third parties frequently became involved to "help" test various implementations and configurations. Only, many of these efforts failed because:

- it became too expensive to outfit labs to perform the tests;
- it became untenable to manage hundreds of permutations (testing and quality assuring dozens of vendors against dozens of other vendor's platforms); or

- it became too time consuming (limited resources in testing labs needed to be scheduled months in advance to perform tests – and tests could also take a lot of time to perform due to configurability issues, latest revision issues, etc.).

Simply stated, Bloor expects these same testing/implementation/configuration issues to rear-their-ugly-heads as Web services interoperability testing ramps-up over the next few years. There will be thousands of implementations of Web services architecture, thousands of possible system permutations, thousands of different applications, and dozens of standard recommendations to be evaluated, tested, and implemented.

Standards Activity

To date, the W3C has been smart enough to avoid getting into the interoperability testing business. It does some interoperability testing in-house to ensure that its standards can be implemented and achieve their goals (observe the W3C Jigsaw Application Server as a proof point of this internal testing). But it knows in the long run that it should stay focused on *standards creation* rather than *standards watchdogging*.

So, if there is little to no W3C standards activity taking place in the area of Web services interoperability standards or testing, how does an IS buyer ascertain whether one vendor's Web services products will interoperate with another vendor's products? And the answer is: this time around the vendor/end user communities have stepped in to create a consortium dedicated to Web services testing. This consortium is known as the Web Services Interoperability Organization (or WS-I).

Vendor/Open Source/Consortia Remedies

How Is Interoperability Regulated/Achieved
The WSI is a voluntary, self-policing interoperability testing organization. It is through this joint and massive testing effort that many industry leading vendors as well as end-user organizations hope to ensure that Web services interoperability is achieved – thus stewarding the success of Web services in the long run.

The original members of the WSI include various computer vendors (note: hardware, business application software, operating systems, and professional services companies are involved) as well as several leading push-the-technology end-user organizations. Some of those members include:

Accenture
Actional
Akamai Technologies
Altova
Autodesk
Avinon
BEA Systems
Borland
Business Objects
Cape Clear Software Inc.
Commerce One
CommerceQuest
Compaq Computer Corp.
Corechange
Corillian Corp.,
Daimler/Chrysler AG
Dassault Systemes
Epicentric
Epicor Software Corp.
ESRI
FileNET
Flamenco Networks
Ford Motor Co.
FrontRange Solutions
Fujitsu
Grand Central Networks
Groove Networks
Hewlett-Packard
IBM
Intel
IONA
Jamcracker
J.D. Edwards
Kana
Loudcloud
Macromedia
McAfee.com
Microsoft
Onyx Software
Oracle
Peregrine Systems
Pivotal Corp.
Plumtree Software
POSC.org
Qwest Communications
Rational Software
RealNames Corp.
Reed Elsevier
Reuters
Sabre Holdings Corp.
SAP
SAS Institute
SilverStream
Sybase
Systinet
Tata Consultancy Services
Tibco
Toshiba TEC Corp.,
United Airlines
Versata
VeriSign,
Vinsurance
webMethods Inc

Source: http:\\\www.WS-I.org

What Bloor Research NA likes best about the WS-I is that it consists of members with a wide range of computer and business experience (as opposed to just systems architects and testing/quality assurance personnel). Member organizations are contributing developers, business analysts, standards architects, marketing personnel – and people from all walks of computing life to the Web services interoperability testing effort.

We also like the fact that these contributors will be developing sample implementation scenarios as well as implementation guideline. These scenarios and guideline should help end users and vendors better understand how to design and build interoperable systems.

Finally, we like the WS-I spirit of commitment to ensure that Web services truly become the first truly interoperable distributed computing standard. The combination of over 500 vendors/end-user organizations involved in setting W3C standards, combined with the commitment of the WS-I membership to achieving cross-platform, cross-operating environment, cross-application language interoperability finally happen creates a powerful force for making Web services the industry's best interoperability architecture.

But do we believe that the WS-I organization will succeed? Based-on past history, we have seen organizations such as this fail for a number of reasons such as funding, resource utilization, and lack-of-market acceptance. We want to see a longer track record of success in resolving vendor interoperability disputes before we declare the WS-I a truly successful endeavor.

Chapter Summary

The W3C, wisely, has not ventured forward into providing interoperability testing for the Web services recommendations that it puts forward. That is not to say that they fail to test their own standards – they do test the standards they recommend internally. They then package a live, working example of a Web services-based interoperability platform (the Jigsaw platform) and make it publicly available as a model for the rest of the industry to follow.

This decision by the W3C not to get into interoperability testing is sage. Interoperability testing has long been a real can of worms – it is oft times expensive to conduct, political in nature, time-consuming, and frustrating. Still, interoperability testing is a necessity in order to verify that various platforms can interoperate.

So, who should do interoperability testing? Should there be a separate, independent firm that conducts interoperability tests and provides certification that certain vendors using certain configurations have successfully proven that they can interoperate with certain other vendors and their respective platforms? This approach has been tried before and has proven to be unwieldy (trying to get systems properly configured and documenting all possible permutations is difficult), costly, and inefficient. Should IS buyers do interoperability testing? (In a way they already do such testing as they seek to integrate ERP, CRM and SCM applications within their own organizations as well as with supply chain partners). But how many IS buyers can afford to shoulder the sometimes-enormous costs associated with interoperability testing?

The right approach seems to be to create a special interest group to provide interoperability testing. This group is populated by vendors as well as IS buyers and end-users – all of whom are interested in achieving cross-platform program-to-program interoperability for their own reasons. And such a group now exists with the creation of the WS-I organization.

At Bloor Research NA we believe that the WS-I offers the right approach to interoperability testing. Vendors now have an interest in achieving interoperability because they believe that it will help sell more distributed systems and software. And users have an interest – interoperability could save some organizations millions of dollars in system and application integration costs yearly. Still, we are somewhat reserved about the long-term success of the WS-I. We need to see a track record of successful resolution of vendor conflicts and implementation issues before we fully endorse the WS-I as the best option for Web services interoperability testing.

PART III: Buyer's Guide – Vendor Selection Criteria/Critique

What This Section Is About

To this point in this report we have spent most of our time learning about what Web services are; what they do; and where shortcomings exist in Web services architecture. But this report has emphasized quite regularly that there exists workarounds for each of the shortcomings/gotchas identified. This section examines some of the many solutions available to fill-in-the-gaps in Web services architecture.

What I Hope You'll Learn in this Section

There are many sources of supply for Web services architectural components. IS buyers can purchase or obtain:

1. Full-function application server environments (including business process integration software, application development environments, portal creation software, personalization software, wireless interconnection software, and much more);

2. Integration servers (designed primarily for interoperability purposes);

3. Point product solutions (point products are narrower in scope than application servers or integration servers – they generally address a single issue like security or management);

These products can be obtained from:

1. Vendors (as expected – from Independent Software Vendors; but there are a fair number of Web services hardware/software appliances making their way to market);

2. Consortia (primarily from various industry organizations); and

3. The open source community; and

4. The standards organizations themselves.

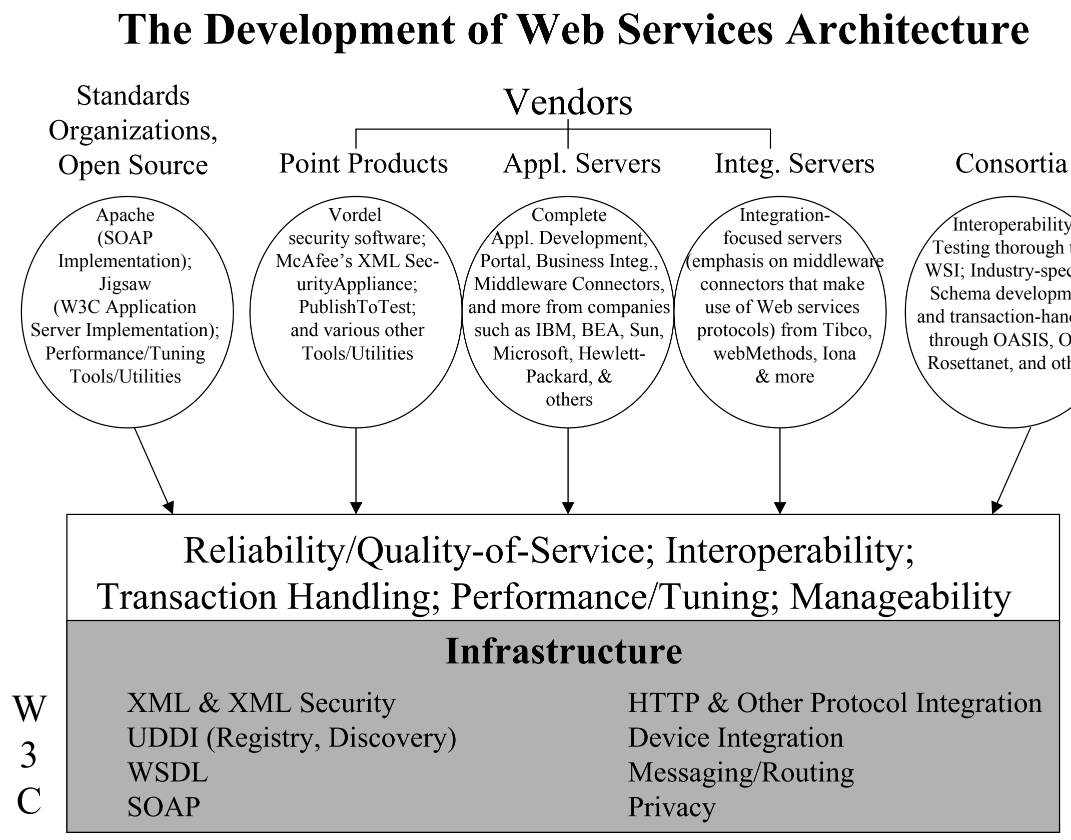
Upon completion of this section we hope that you, the reader, has a solid understanding of how Web services software is packaged and providing information on various sources of supply.

Chapter
9

Sources of Supply for Web services Architecture, Products, and Services

Throughout this report we have emphasized that Web services has shortcomings – and that most of those shortcomings can be overcome using various point product solutions or other more complete solutions provided by various Web services architecture/component suppliers. Now its time to prove this point by providing a high-level overview of various, randomly chosen Web services product offerings that help solve problems such as reliability, security, scalability, manageability and the like (see Figure 13 – below).

Figure 13 – Contributors to the Development of Web Services Architecture


The Development of Web Services Architecture
Standards Organizations, Open Source
Vendors
Point Products
Appl. Servers
Integ. Servers
Consortia
Apache (SOAP Implementation); Jigsaw (W3C Application Server Implementation); Performance/Tuning Tools/Utilities
Vordel security software; McAfee's XML SecurityAppliance; PublishToTest; and various other Tools/Utilities
Complete Appl. Development, Portal, Business Integ., Middleware Connectors, and more from companies such as IBM, BEA, Sun, Microsoft, Hewlett-Packard, & others
Integration-focused servers (emphasis on middleware connectors that make use of Web services protocols) from Tibco, webMethods, Iona & more
Interoperability Testing thorough the WSI; Industry-specific Schema development and transaction-handling through OASIS, OBI, Rosettanet, and others
Reliability/Quality-of-Service; Interoperability; Transaction Handling; Performance/Tuning; Manageability
Infrastructure
W 3 C
XML & XML Security
UDDI (Registry, Discovery)
WSDL
SOAP
HTTP & Other Protocol Integration
Device Integration
Messaging/Routing
Privacy

Source: Bloor Research North America, May 2002

How Is This Chapter Organized?

First, this chapter discusses the differences between application servers, integration servers, and point products.

Second, Bloor Research NA provides a list of Web services product and service providers. This list consists of:

1. Vendor offerings;
2. Consortia offerings; and
3. Open Source/Standards Organizations offerings.

After the company (or consortium name) is a brief description of the Web services product offered, followed by Bloor Research NA comments on the particular vendor or product. This vendor/open source/consortia section is organized by provider (vendor, open source, etc.); then followed by a brief product description; and then concluded with Bloor Research NA commentary. These short vendor/product-/commentary descriptions are designed to help your organization understand what products and services are available to help your organizations fill-in-the-holes in Web services architecture such that you can proceed to build mission-critical capable Web services applications.

Hopefully, armed with these representative samples of products and services, your organization will be able to what products are available in the marketplace in order to help jumpstart your organization's move into Web services architecture.

Application Servers, Integration Servers, and Point Products

Bloor Research NA separates vendor Web services offerings into three categories:

1) Application servers;
2) Integration servers; and,
3) Point products.

Contrast: Application Servers Versus Integration Servers

Application servers are very complete Web platform environments. They generally contain highly-integrated software that provides XML creation/security, business process management, application development, portal development, personalization, wireless connection and a myriad of other Web platform-related value-added software – all of which run on top of Web services protocols and work with Web services UDDI registries.

Integration servers are like application servers in some respects. Application servers frequently have software that allows for connection between Web services applications and legacy custom or packaged applications. So do integration servers. Application servers frequently provide tools for application design, development and deployment. So do some integration servers. The primary difference can be found in where application server vendors focus versus where integration server companies focus.

For the most part, application servers are all about capturing the hearts and minds of the application development community. Application servers emphasize providing tools and utilities that make it easy for application developers to build and deploy applications (and closely link those applications to business logic/business flow processes). Application server makers hope that by making the application development job extremely easy, it will be hard for enterprises to switch to other competitive platforms.

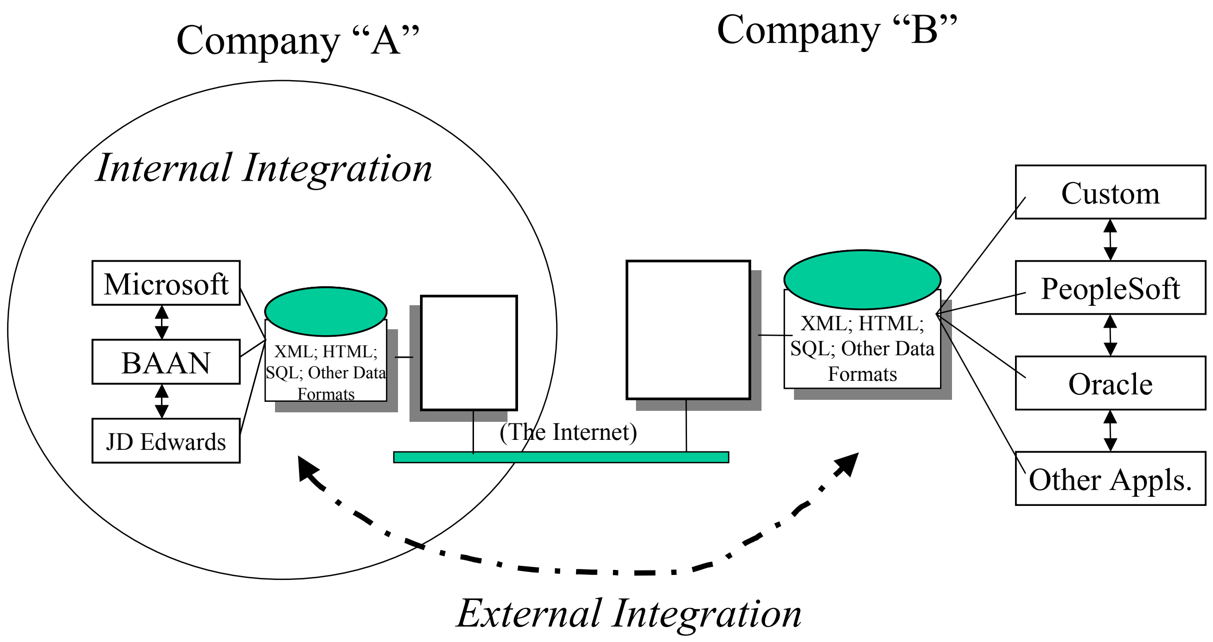
Integration servers, for the most part, focus on providing connectivity and interoperability between various applications environments. So integration servers focus on providing "connectors" or "adaptors" that enable businesses to tie together existing packaged/custom applications, and or to tie together existing/custom applications with new Web services applications. Integration servers can usually be found in enterprises looking to integrate internal disparate applications and databases or in that are looking to interoperate with applications and databases that reside within the information systems infrastructure of supply chain business partners (see Figure 14 – next page).

Their primary differences between application servers and integration servers is *emphasis*:

- Integration servers emphasize inter-enterprise and extra-enterprise *interoperability.* In some cases, application servers also provide connectors and adapters that allow for inter/extra-enterprise integration.

- Application server vendors (such as HP, IBM, Microsoft, Sun, et al) emphasize *application development*. These vendors provide tools for designing, developing, and deploying Web services applications. But they also provide tightly integrated add-on software that allows for Web services/business process integration, portal development, wireless integration, personalization – and a whole lot more. Application server makers provide highly-integrated ancillary programs in the hope that buyers will develop a strong preference for a given application server as a development platform – and preference will drive recurring platform sales.

Figure 14 – How Integration Servers Are Used



Integration server vendors focus on providing connectors/adapters that help enable application-to-application interoperability. This can be within an enterprise – or across enterprises (integrating applications with supply chain business partners).

Source: Bloor Research North America – May, 2002

Point Products
Point products are simply software (or sometimes hardware/software appliances) that serve to address a single issue such as security or manageability. In the following section over a dozen examples of point products are described. These products range from tracking and billing systems (such that a Web service provider could charge a per use or license fee for Web services provided) through point products for security, or management, or even to put together a library of application objects that could be used to construct Web services applications.

Why would an IS administrator, system architect, or IS manager want to purchase a point product? Point products serve to fill-in-the-gaps that other platforms may not yet address. For instance, a point product could be used to provide a management console for tracking and monitoring the status of a Web services application. Or a point product could be used to provide document/data level security. Or a point product could be used to help manage a group of Web services developers. And so on.

Generally speaking, point products are used to augment a server or operating environment. For instance, an IS manager may have downloaded a copy of the Apache Web services application server – but may want better line or document security. Point products could be used to provide such security.

Vendor/Open Source/Consortia Offerings

Notes: (1) Some vendors are mentioned in several categories due to multiple product offerings. (2) Some products mentioned do not specifically use Web services protocols or the registry – but can be used to augment Web services architecture. (3) This listing is not intended to be comprehensive, just representative.

Systems/Application Management Providers

Vendor Name	Product/Service Offering	Bloor NA Commentary
Actional ***www.actional.com***	**Centralized workstation management. Publishing; security; management of Web services.**	**Fills several shortcomings in Web services architecture – particularly manage-ment. Close relation-ship with Microsoft .**
Allesta ***www.allesta.com***	**A "Web services Agency". Provide Web services monitoring and infrastructure-related services as well as hosted services.**	**Very much like an application solutions provider. This company will help you manage and monitor your Web services environment.**
Altoweb ***www.altoweb.com***	**Deployment, testing, and management (monitoring) of Web services.**	**Can be used as a "point product" add-on to BEA WebLogic and IBM WebSphere.**
Collaxa ***www.collaxa.com***	**Build Java scalable applications. Administer, audit, and monitor business activities.**	**Emphasis on orchestrating loosely coupled Web services environments.**
Flamenco Networks ***www.flamenco.com***	**Web services "connection provisioning". Security, Management, performance.**	**Web services viewed from a how-to architect the network perspective.**
Grand Central Communications ***www.grandcentral.com***	**Connection, security, orchestration, management, monitoring, and messaging.**	**Helps to establish a reliable Web services environment for B2B information exchange.**

West Global ***www.westglobal.com***	**Enterprise management with the ability to tune performance.**	**Interesting mix of manageability and performance tuning.**

Note: IBM, HP, Sun Microsystems, Microsoft and application server vendors offer integrated Web services manageability products. See application server category for further details.

Device Interconnection Providers

Vendor Name	**Product/Service Offering**	**Bloor NA Commentary**
Alertwire **www.alertwire.com**	**Provides mobile access to existing computing environments for various devices.**	**Somewhat addresses need for multi-device connection.**
MagnetPoint **www.magnet point.com**	**A library of Web services connectors as well as various device support.**	**Worth a look for connecting disparate applications together; and for device support.**

Note: IBM, Sun Microsystems, Microsoft, and other large application server vendors also offer device interconnection products as part of their application server platform offering.

Security Providers

Vendor Name	**Product/Service Offering**	**Bloor NA Commentary**
Actional ***www.actional.com***	**Management and security products.**	**Fills several shortcomings in Web services architecture – particularly management. Close relationship with Microsoft & .NET.**
CommerceQuest	**Protect MQ**	**MQSeries-based security enhancer.**
IBM ***www.IBM.com***	**Tivoli is a complete security and management environment that is integrated with IBM's WebSphere server.**	**Go to IBM Web site for further details.**

Netegrity ***www.netegrity.com***	**Security, management of user profiles, personalization.**	**Point product solution for security and management of users.**
Oblix ***www.oblix.com***	**Identity; single systems log-on; and access control software. Also, process management.**	**Identity management – could be useful for both business and personally-oriented Web services applications.**
RSA Security **www.RSAsecurity .com**	**Encryption, public key, and other security products.**	**Network level as well as application level security.**
Stele, Inc. **www.stele.com**	**Centralized security, manageability, reliability.**	**Web services infrastructure provider.**
Vordel ***www.vordel.com***	**Web services security.**	**An excellent example of a Web services security point product.**
Verisign **www.verisign.com**	**Security, key encryption.**	**Working with IBM, Microsoft, and other vendors on public and private key encryption for Web services.**

Note: Sun Microsystems, Microsoft, and other large application server vendors also offer device interconnection products as part of their application server platform offering.

Performance Tuning Providers

Vendor Name	**Product/Service Offering**	**Bloor NA Commentary**
CommerceQuest	**Test suite that works with Load Runner.**	**Tests for MQSeries environments.**
Flamenco Networks ***www.flamenco.com***	**Web services "connection provisioning". Security, Management, performance.**	**Web services viewed from a how-to architect the network perspective.**
Kenamea	**The company sells development**	**A complete archi-tecture geared at**

www.kenamea.com	**environments and a message switch software that provides for reliability, security, connectors, adapters, plus performance tuning.**	**addressing reliability and performance shortcomings.**
Push-to-Test ***www.pushtotest. com***	**Intelligent agents to test performance, scalability, and the network.**	**Open source code.**
West Global ***www.westglobal.com***	**Enterprise management with the ability to tune performance.**	**Interesting mix of manageability and performance tuning.**

Other Web services-related Providers

Vendor Name	**Product/Service Offering**	**Bloor NA Commentary**
Am-beo ***www.am-beo. com***	**Rating/revenue metering and billing system.**	**A way to handle billing and micro-transactions for Web services.**
Documentum ***www.documentum .com***	**Content platform and repository.**	**Create and manage XML, documents, other files using one common repository.**
LogicLibrary ***www.logiclibrary. com***	**Product is a repository for object program assets. (Not UDDI – but a place where assets can be published and found).**	**Helps track assets for later application assembly and development.**
MetraTech ***www.metratech. com***	**Web services billing, revenue sharing and settlement software.**	**Potential to be used for pay-as-you-go services or in software-as-a-service environments.**
OpenLink ***www.openlink sw.com***	**Middleware needed to integrate databases, file servers/systems, web sites, etc.**	**Runs on multiple platforms – including Linux.**
Sonic Software	**Integration, messaging server environment.**	**Plus professional services.**

www.sonicsw.com		
Stratify *www.stratify.com*	Put unstructured data into UDDI directory.	Exciting way to get into UDDI today.
Spotfire *www.spotfire.com*	Business intelligence.	Useful with WSDL over time in order to create reports from various business applications.
Talaris *www.talaris.com*	A Web services-based exchange. Broker, procure, manage, deliver personalized service, …	Designed to attack electronic procurement problem using Web services.

Reliability/Transaction Handling Providers

Vendor Name	Product/Service Offering	Bloor NA Commentary
Blue Titan www.bluetitan.com	Reliability, management, and scalability focus. Also offer Web services application development tools.	Result of merger of Velocigen & Service Mesh. Excellent way to approach making Web services reliable.
Grand Central Communications	Connection, security, orchestration, management, monitoring, and messaging.	Helps to establish a reliable Web services environment for B2B information exchange.
Hewlett-Packard	Reliability features can be found in operating system and other places such as networking or with trans-action monitor software or other vendor-created products.	Special mention: working closely with OASIS; integrating Business Transaction Protocol.
IBM	Reliability features can be found in operating system and other places such as networking or with trans-action monitor software or other vendor-created	See application server vendors.

	products.	
Kenamea ***www.kenamea.com***	**The company sells development environments and a message switch software that provides for reliability, security, connectors, adapters, plus performance tuning.**	**A complete archi-tecture geared at addressing reliability and performance shortcomings.**
Microsoft	**Reliability features can be found in operating system and other places such as networking or with trans-action monitor software or other vendor-created products.**	**See application server vendors.**
SilverStream **www.silverstream .com**	**A complete application server environment.**	**Strong emphasis on reliability. HA cluster with no single point of failure. Scalable, manageable. Monitor plus performance tuning features.**
Sun Microsystems	**Reliability features can be found in operating system and other places such as networking or with trans-action monitor software or other vendor-created products.**	**See application server vendors.**
The Mind Electric **www.themind electric.com**	**GAIA grid server to load-balance, fail-over, and cluster.**	**Leading edge approach to reliability.**
Zebrazone ***www.zebrazone. com***	**"Eternal Availability". A massively distributed architecture that works on IBM, Sun and HP platforms.**	**One of the more intriguing approaches to building reliable Web services environments.**

Application Server Providers

Vendor Name	Product/Service Offering	Bloor NA Commentary
Apache *www.apache.org*	Open source application server environment. Just the basics – a free, nuts-and-bolts basic application server. Platform consists of the basic Web server protocols (UDDI, SOAP, WSDL).	A good place to turn for a basic platform. IS buyers can then add point products to create their own application server environments or just use for basics.
BEA *www.BEA.com*	Complete application server environment. Have built product line largely by integrating other vendor's products (acquisition).	One of the most popular Web server environments. Good partnerships with professional services suppliers; good rela-tionships with vendor who add applications.
Bowstreet *www.bowstreet.com*	Solid application server environment (complete with portal and application development environment).	Works with BEA and on other platforms. Also provide education and consulting services.
Hewlett-Packard *www.hp.com*	Comprehensive approach with the "Netaction" product suite.	Unique approach to transaction handling.
IBM *www.ibm.com*	WebSphere product suite. Visit WebSphere site for further details.	One of the most comprehensive product sets in the industry. Plus largest IT services group.
IONA *www.IONA.com*	Strong integration connectors, but also offers application development, management, & security.	Straddles both the integration server market and the application server market.
Microsoft *www.Microsoft.com*	Go to .NET on www.Microsoft.com.	Application develop-ment environment clearly targeted at capturing the hearts and minds of develop-

		ers. Outstanding integrated product set.
Oracle ***www.oracle.com***	**Solid application development environment. Good tools for integrating database w/Web services.**	**Late to market; confused marketing effort.**
Polarlake ***www.polarlake.com***	**Application development tools and utilities, web services, portals, the whole enchilada.**	**Comparatively inexpensive add-ons beyond the basic platform.**
SilverStream **www.silverstream .com**	**A complete application server environment.**	**Strong emphasis on reliability. HA cluster with no single point of failure. Scalable, manageable. Monitor plus performance tuning features.**
Sun ***www.sun.com***	**Go to Sun ONE on Sun site.**	**Founder/keeper of Java development language.**
Sybase ***www.sybase.com***	**Web services application development environment plus other Web server building/integration tools.**	**Long known for excellence in application development and databases.**
W3C **www.w3c.org**	**Jigsaw application server.**	**Excellent open source code implementation of W3C standards.**

Note: Most application server product sets are too expansive to detail in this report. The names of vendor Web services application server platforms have been included here – readers are encouraged to go to respective vendor sites to garner detailed information.

Integration Server Vendors

Vendor Name	**Product/Service Offering**	**Bloor NA Commentary**
Attachmate ***www.attachmate***	**Connectors/adapters that enable various applications to interoperate.**	**Many adapters for all sorts of system environments including older proprietary**

.com		systems as well as new open systems.
CommerceQuest	Complete integration and business process environment.	Solid MQSeries-based architecture.
Commerceroute *www.commerce route.com*	An integration "appliance". Turnkey support for multiple XML flavors.	Especially useful in business-to-business environments.
Tibco *www.tibco.com*	Connectors/adapters for integrating Web services with legacy applications.	Outstanding products – proprietary network bus being migrated to Web services architecture.
webMethods *www.webmethods .com*	Web services integration connectors/adapters.	Understands very well how to link disparate applications together using WSDL.

Web Services Application Development/Process Integration Vendors

Vendor Name	Product/Service Offering	Bloor NA Commentary
Avinon *www.avinon.com*	Visually-oriented application development environment for Web services.	Easy-to-use build-your-own Web services environment.
Bind Systems *www.bindsys.com*	Expose business processes as Web services.	Highly-useful tool for understanding process flow and design. Helps streamline business operations.
Bizconverse www.bizconverse-.com	Business-to-business integration products. Define, build, modify, and inspect XML schema.	Helps enterprises build Web services applica-tions that can be used to interface with business partners.

Blue Titan *www.bluetitan.com*	Reliability, management, and scalability focus. Also offer Web services application development tools.	Result of merger of Velocigen & Service Mesh. Excellent way to approach making Web services reliable.
Borland *www.borland.com*	Expansive offerings. Provide Web services application development tools.	Strong Java, .NET application development tools. Plus Linux support.
Business Objects *www.businessobjects .com*	Suite of applications that allow for decision support/ business intelligence drill down.	Can be used to garner information from disparate systems to create reports.
Bizconverse www.bizconverse-.com	Business-to-business integration products. Define, build, modify, and inspect XML schema.	Helps enterprises build Web services applica-tions that can be used to interface w/ business partners.
Cacheon *www.cacheon.com*	J2EE application creation and migration.	Tool for migrating Java applications from one platform to a different platform.
Collaxa *www.collaxa.com*	Build Java scalable applications. Administer, audit, and monitor business activities.	Emphasis on orchestrating loosel-coupled Web services environments.
Cypresslogic www.cypresslogic.com	Graphically-oriented J2EE and .NET development tools.	User friendly – develop Web services without coding.
Infragistics *www.infragistics.com*	Java, .NET development environments.	Visually-oriented to make programming easier for developers.
Infravio *www.infravio.com*	Web services design, management, and execution.	A Web services design and execution environment.
Iopsis Software *www.iopsis.com*	Creation, assembly, deployment, and publication of Web	Graphically-oriented.

	services.	
Kinzan *www.kinzan.com*	Application development tools plus process integration.	A Web services design and execution environment.
Macromedia *www.macromedia.com*	Very popular suite of Web application development tools and utilities.	A market leader – with good reason.
Orchestra Networks *www.orchestra networks.com*	Application development and management for Web services environments.	"Orchestrates" Web services application development and process flow.
Primordial *www.primordial.com*	Web services "proxy" server. Provides application development, process harmonization, and security.	A Web services design and execution environment.
Rational *www.rational.com*	Application development, system testing, and collaborative project management.	The project management aspect really stands out. Good team development tools.
Red Gate *www.redgate.com*	Application development tools for Microsoft .NET environments.	One of many.
Rogue Wave *www.roguewave.com*	C++ application development environment for Web services.	Application development plus professional services consulting.
SoftQuad *www.softquad.com*	XML development tools.	One of many.
Systinet *www.systinet.com*	Scalable, secure Web services application development.	One of many.
Togethersoft *www.togethersoft. com*	Web services application development environment.	One of many.

Versata ***www.versata.com***	**Business logic plus services.**	**Close ties with BEA and IBM.**

Web Services Professional Services Suppliers

Vendor Name	**Product/Service Offering**	**Bloor NA Commentary**
Persistent Web ***www.persistent.com***	**Web services design and deployment guidance – particularly as it relates to eBusiness.**	**A place to turn for Web services and eBusiness consulting for the Web.**
Valtech ***www.valtech.com***	**Business consulting and professional services for Web service deployments.**	**Close ties to BEA.**
WebGain	**Java application development environment.**	**One of many.**

Chapter Summary

There are three sources of supply for Web services products and services:

1. *Individual "point" or open source products* – these products are self contained modules that provide developers with access to the building blocks needed to build a Web services infrastructure (for instance: SOAP protocol can be acquired in open source form by downloading the Apache SOAP open source protocol). Point products would include a security solution such as those provided by McAfee with its WebShield security/antivirus appliance or by Forum Systems with their content security appliance products). These products would be used by enterprises that have a need (or prefer to) build Web services solutions from "scratch".

2. *Integration servers* – there are several companies that have evolved from the enterprise application integration (EAI) marketplace that now provide Web service connectors. The way this works is that various integration server companies have developed "connectors" or "adapters" for custom or packaged application (such as SAP, Oracle, PeopleSoft, and others) – enabling these applications to work cooperatively with each other. These adapters are now being reworked to "speak" Web services (SOAP and WSDL) – thus attacking application interoperability issues using Web services protocols as a default way to resolve communications and program-to-program communications issues. (Also note that integration

server companies frequently specialize on other aspects of program-to-program communications – for instance: security, manageability, or reliability. A good example of this is webMethods – a company that offers a wide range of packaged application adapters, but also offers excellent application management tools and utilities). Companies that fit into this market segment include Tibco, webMethods, Iona, and a few dozen others. These products would be used by enterprises that have a need to solve program-to-program interoperability problems; or enterprises that are seeking to integrate new Web services applications with existing packaged or custom applications.

3. *Application servers* — these server environments focus on far more that the applications integration and interoperability issues (that are addressed by integration server vendors). Application servers include a wide variety of "add-on" tools, utilities, and programs that help enterprises build, deploy, secure, optimize, integrate, and manage servers that are used. These integrated server environments frequently include products that help IS managers:

    - develop applications
    - build Web servers (portals);
    - integrate business process management;
    - add voice command structures;
    - integrate wireless communications;
    - integrate office applications; and
    - drive all of these integrated applications over a Web services architecture.

    These products would be used by enterprises that want a highly integrated application development environment complete with business integration tools, Web development tools, and other elements that make it possible to build robust, reliable, secure server environments.

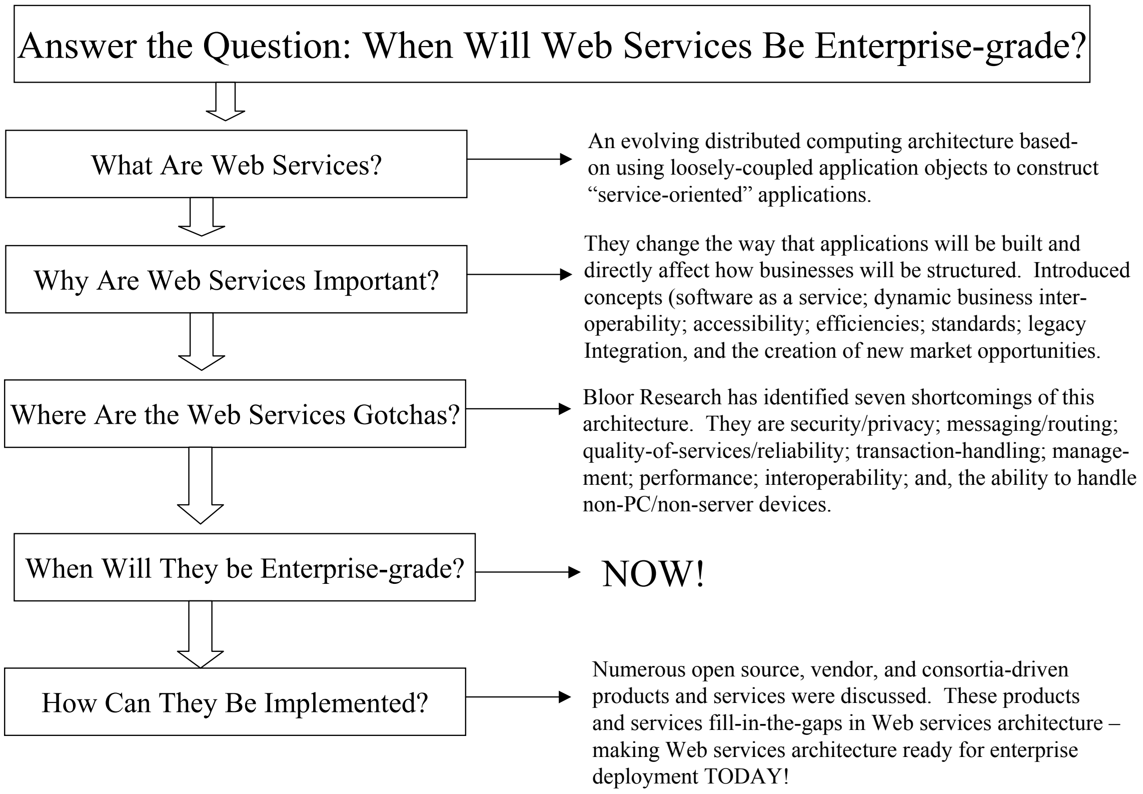
PART IV: Conclusion

Summary Observations

The Logical Flow of This Report

This report has sought to provide you with an understanding of when Web services architecture will be robust enough to be used in mission-critical computing environments. To answer this question we organized the report as follows (see Figure 15).

Figure 15 – The Logical Flow of this Report


Answer the Question: When Will Web Services Be Enterprise-grade?
What Are Web Services?
An evolving distributed computing architecture based-on using loosely-coupled application objects to construct "service-oriented" applications.
Why Are Web Services Important?
They change the way that applications will be built and directly affect how businesses will be structured. Introduced concepts (software as a service; dynamic business inter-operability; accessibility; efficiencies; standards; legacy Integration, and the creation of new market opportunities.
Where Are the Web Services Gotchas?
Bloor Research has identified seven shortcomings of this architecture. They are security/privacy; messaging/routing; quality-of-services/reliability; transaction-handling; manage-ment; performance; interoperability; and, the ability to handle non-PC/non-server devices.
When Will They be Enterprise-grade?
NOW!
How Can They Be Implemented?
Numerous open source, vendor, and consortia-driven products and services were discussed. These products and services fill-in-the-gaps in Web services architecture – making Web services architecture ready for enterprise deployment TODAY!

Source: Bloor Research North America – May, 2002

The Three Most Important Concepts in This Report

As stated in the *Preface*, there are three really important messages that this report delivers:

1. *Web services are distinctly different from predecessor distributed computing architectures due to the ability to loosely-couple applications*. This loose-coupling results in new ways to link applications together to more quickly build applications; to create or respond to competitive pressure more nimbly; to bring products to market more efficiently; and so on. It also removes a lot of human programmer involvement as applications perform services for other applications without programmers having to "hard-wire" applications together.

2. As a result of the ability to loosely-couple applications, businesses will find new ways to architect their information systems. *This new architecture will change the way businesses operate and will cause fundamental business models to change.*

3. *Web services architecture has some maturing to do* if it is to be used in mission-critical computing environments. The way that we illustrate this point is to examine where standards committees are spending their time and what they are spending their time on. By so doing, we are able to identify where Web services architecture needs improvement – and we're also able to determine when a particular standards committee is likely to make a "recommendation". This approach, we believe, will enable readers of this report to determine when implementing Web services will be right for their own organizations.

How We Reached Our Ready-for-Prime-Time Conclusion

Bloor Research NA is aware that there have been a number of studies conducted by other research organizations that indicate that it will be years before this architecture is enterprise-grade (mission-critical capable). But our research indicates quite the opposite – we have:

- Closely examined the standards activity that takes place at the W3C;
- Considered what it takes to build a robust distributed computing environment;
- Identified seven shortcomings (gotchas) in the architecture;
- Found solutions for everyone of the shortcomings that we identified;
- Talked to users (not included in this report – part of "Web Services Explained" by Joe Clabby – Prentice Hall, July 2002); and
- Concluded that Web services architecture can be deployed now in mission-critical computing situations.

Web services architecture has a lot of maturing to do. The W3C has focused (and rightfully so) on finding ways to address network and content security; to address the concept of services discovery; to handle routing/messaging issues; on interoperability; and to integrate wireless devices with Web services architecture. But efforts to ensure security, reliability, to improve transaction handling, to improve application manageability, and address performance/tuning issues still need to be addressed in future standards work.

Having said this, it is the observation of Bloor Research NA that the vendor community is actively and aggressively moving forward to build products and to offer services that help to address Web services architectural shortcomings. New Web-services-enabled products such as security appliances that can also handle increased routing/message handling requirements are starting to come to market (see Forum Systems and McAfee write-ups in the final part of this report for examples of such appliances). New products that take advantage of W3C recommendations for public key encryption plus add manageability features are also coming to market (see Cyclone Commerce Inc. write-up in the vendor section of this report). Other applications manageability products that can manage Web services environments are being brought to market by IBM (with its Tivoli product set, while webMethods offers its own manageability solutions for Web services environments. Hewlett-Packard now offers specialized software for Web services "transactioning" (handling complex transactions within a Web services environment"; while Sun, BEA, Microsoft, and others offer similar extensions to their own respective product lines.

When and How Will Web Services "Mature"?

These are the three key questions that this report sought to answer because by understanding the answers to these questions enterprise IS buyers will be able to better plan for Web services application development and deployment.

When?
With respect to the question: "when will Web services be mature enough to be used in mission-critical, high-transaction-oriented environments" – the quick answer is "NOW". Note that the W3C is aggressively pursuing the development of standards for security, routing/messaging, interoperability, and multi-device support. The W3C has already released a long list of standards recommendations for XML security and has done a lot of work to ensure that non-PC/non-server devices (for instance, wireless devices) are capable of participating as full-fledged citizens on the Web. Then combine W3C standards work with efforts being made by vendors, consortia, other standards organizations and the open source community (found in Chapter 9) and the composite picture shows that Web services architecture is fast maturing.

How?
It is our belief that we will see Web services architecture mature in the following way:

- *Maturation through continued definition and ongoing standards activity* – The basic architecture will continue to be defined (until mid-2003) through the workings of the Web Services Architecture Group. (A definition of this group's charter can be found at http://www.w3c.org/2002/01/ws-arch-charter). In short, this group is the organization that considers how to make Web services modular, platform independent, extensible, and so on. Other activities and groups will also contribute to the advancement and maturation of Web services architecture – particularly several XML-related standards setting activities and work groups.

- *Maturation over time* – Specific elements of Web services (such as the SOAP and WSDL protocols, and the UDDI registry service) have already been defined – but will continue to be honed over the next several years. (For instance, much work remains to be done on UDDI discovery capabilities; on SOAP security; and other aspects of these protocols/registry) So, time and continuing efforts to refine and hone Web services architecture will also help this architecture mature.

- *Augmenting Web services architecture using open source or vendor products* – New "recommendations" (the W3C term for "standards") are forthcoming over the next few years in the areas of security, routing/messaging, interoperability, and non-PC/server device adaptation to Web services. Meanwhile, other activities are not yet slated for standardization (such as manageability and reliability). So, Bloor Research NA expects to find various open source contributors as well as vendors filling-in-the-gaps in Web services architecture in lieu of standards – until such times as standards become available.

  Examples:

  - IBM's Tivoli can be used to help provide security for Web services environments;

  - WebMethod's management console and interoperability "adapters" products can be used to help manage Web services applications establish interoperability between disparate application/database environments; and,

  - Microsoft's existing performance analysis and tuning products can be used to tune Web services application servers in order to optimize performance.

- *Through the efforts of industry consortia* – A Web Services Interoperability Organization has been formed to help work out vendor interoperability issues that may occur as vendors interpret and implement Web services standards. Rosettanet, OBI, and OASIS (all described later) all contribute to helping build XML schema for various industries – and these schema will be heavily used over Web services architecture (when the architecture matures a bit).

In summary, Web services architecture is maturing quickly thanks to the efforts of the W3C combined with contributions from the vendor, consortia, and open source communities. The W3C is focused on security, routing/messaging, interoperability, and device independence; the other organizations are also bringing products to market that either use the W3C recommendations or augment them with additional features. And, the vendor/consortia/open source communities are also focusing on quality-of-service/reliability; transaction handling, manageability, and performance/tuning – areas that must be enhanced in order to enable Web services architecture to be used in mission-critical computing environments. The ultimate answer to the question: "when will Web services be robust enough to be used in mission-critical computing environments" is (depending on the enterprise's requirements for security, reliability, etc.) quite possibly "now".

What Else Is Needed?

Bloor Research NA notes that the vendor and open source communities as well as various consortia have graciously stepped forward to fill-in-the-gaps in Web services architecture. But, have said this, we also observe that there is a need for vendors, service providers, and enterprise users to create *best practices recommendations* for the deployment, management, and tuning of Web services applications. We have found some of this type of information through the W3C, and by scouring various user group forums for Web services experiential data. But for enterprises to better exploit Web services – and for Web services to really take-off – best practices information on deployment, management, and tuning needs to become more readily available.

Final Statements

One reviewer of this report suggested that it include a discussion about how Web services can be applied in real-world computing environments. Another suggested that the vendor section be expanded to include more insight into which vendors do what — and how they can be differentiated. To address these comments, Bloor NA suggests that interested readers examine Joe Clabby's new book ("Web services Explained", Prentice Hall, July, 2002). Mr. Clabby is president of Bloor Research – North America, and one of the primary authors of this report.

Again, we at Bloor NA would like to thank IBM for purchasing Web distribution rights for this report. We think that IBM has done the computing industry a valuable service by helping to surface some of the shortcomings of Web services architecture and by encouraging a conversation about how these remedies can be overcome. We believe that by so doing, IBM is furthering the understanding of Web services architecture — a move that will lead to more rapid adoption of Web services in computing environments in the near term.