Technology Advice.
Business Results.
Giga Information Group®

The State of Web Services

Uttam Narsu
Vice President

November 14, 2002

© 2002 Giga Information Group, Inc. All rights reserved.
Reproduction or redistribution in any form without the prior written permission of Giga Information Group is expressly prohibited.

Initially hyped, now anti-hype, yet:

- Government interest remains strong
- Businesses merely deferring expectations of its value

Recent CIO surveys:

- Application integration (including Web Services) #2 priority behind security
- 34% already have Web Services strategy in place with 28% planning on implementation in next two quarters

Key Concept: The Service

Reply (to Request)
- **Message**
- **Transaction**
- **Business Unit of Work**

Request
- **Message**

Service Interface

Legacy Apps/ Components

New Services

New Components

© 2002 Giga Information Group, Inc.

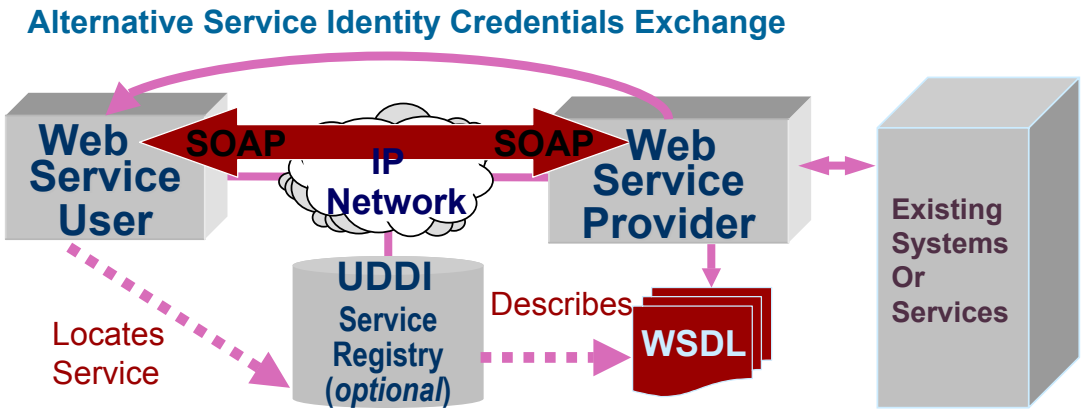
When you go to the restaurant, you don't want to have to instruct the fry cook to make the chips or French Fries, or the burger-flipper to make the burger. You want a convenient layer of service-interaction "glue" as provided by the person staffing the drive-through window, to handle those details for you.

Well designed software services will offer this same kind of glue, convenient syntactic sugar to ease usage of the service, and also taking into account the need for packaging and aggregation of lower-level application calls, components, and interfaces. Additional design goals include optimizing for performance over slower network links with longer latency (not good for chatty component interface invocations), integration into an implicit workflow with known state, and enforcement of quality-of-service agreements that need to be monitored and controlled.

What Is a Web *Service*?

A *Web Service*:

- **Has a Service interface implemented with SOAP/XMLP**
- **With SOAP/XMLP likely implemented over Internet protocols like HTTP and/or SMTP**
- **With the interface described using WSDL**
- **With the service registered/discoverable via UDDI, if at all**



© 2002 Giga Information Group, Inc.

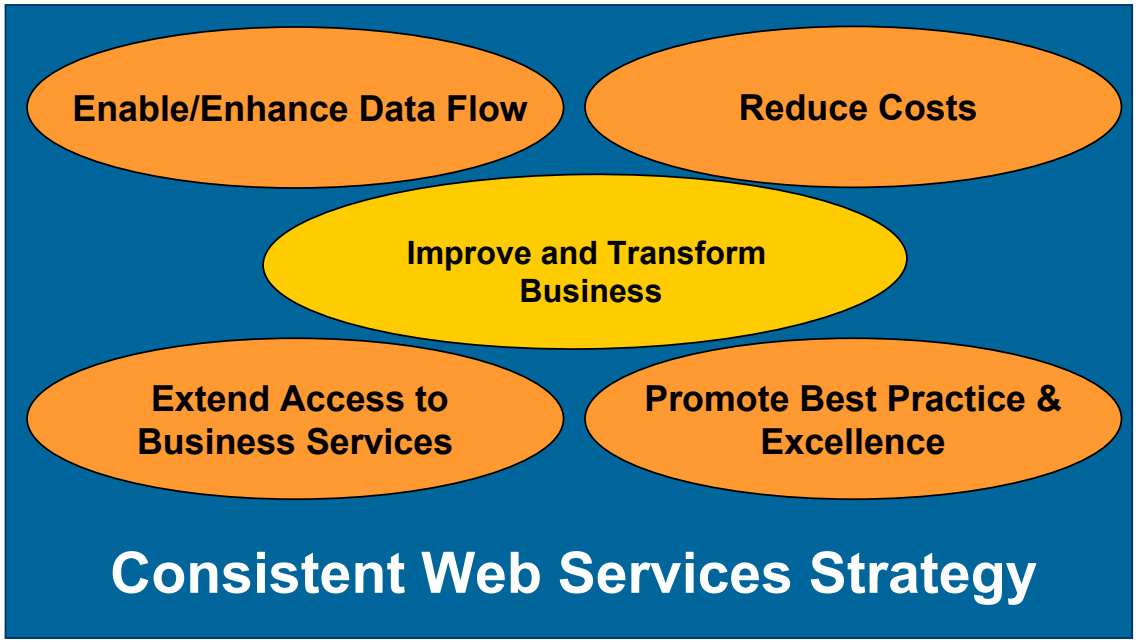
So, a Web Service is the specific implementation of this concept using Simple Object Access Protocol (SOAP), which may be called XML Protocol by the W3C, where SOAP is also likely to be implemented over Internet protocols like HTTP and/or SMTP. Yet this is not a requirement, SOAP can still be SOAP and use some other protocol underneath.

A Web Service interface is described using Web Services Description Language (WSDL), which is a form of XML document. And if the Web Service is to be registered and discoverable, this is done via the Universal Description, Discovery, and Integration (UDDI) interface. But even in the specific Web Services context, some Web Services are either registered only in private registries accessible only to one company or on an extranet to a specific group of trading partners, or not registered at all, with credential exchanged in some other way, even as simple as e-mailing a WSDL file and URL to call to a trading partner after they have signed up for the service.

A definition of a Web Service is:

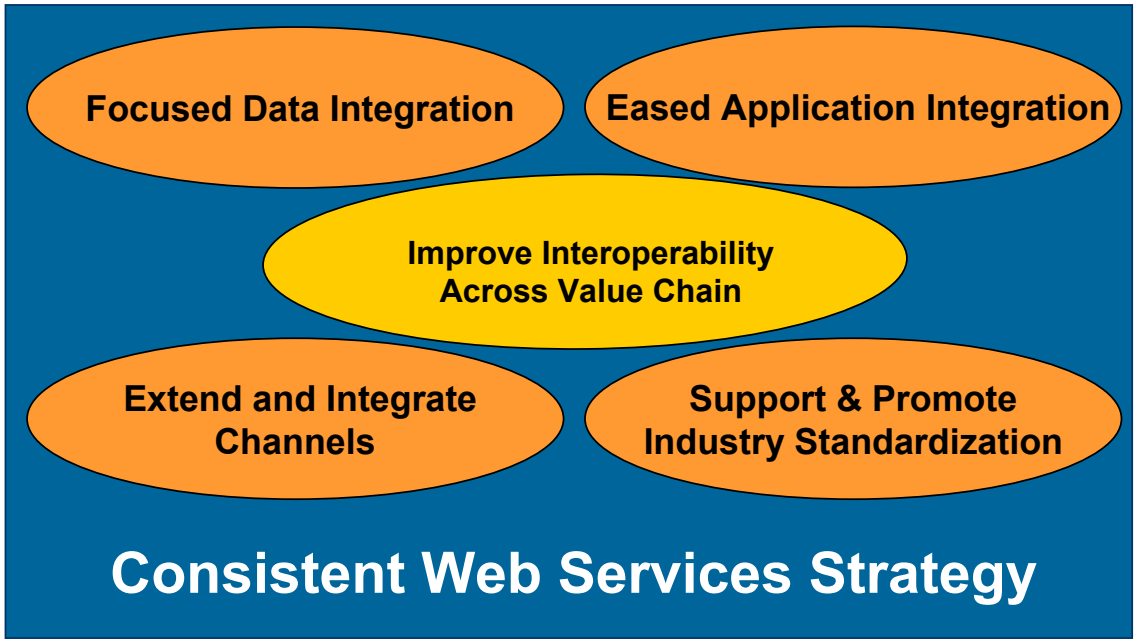
> A Web Service is an interface implemented by one or more applications or components that provides one or more business services to other applications or end-users via standardized Web protocols.

Business Benefits



© 2002 Giga Information Group, Inc.

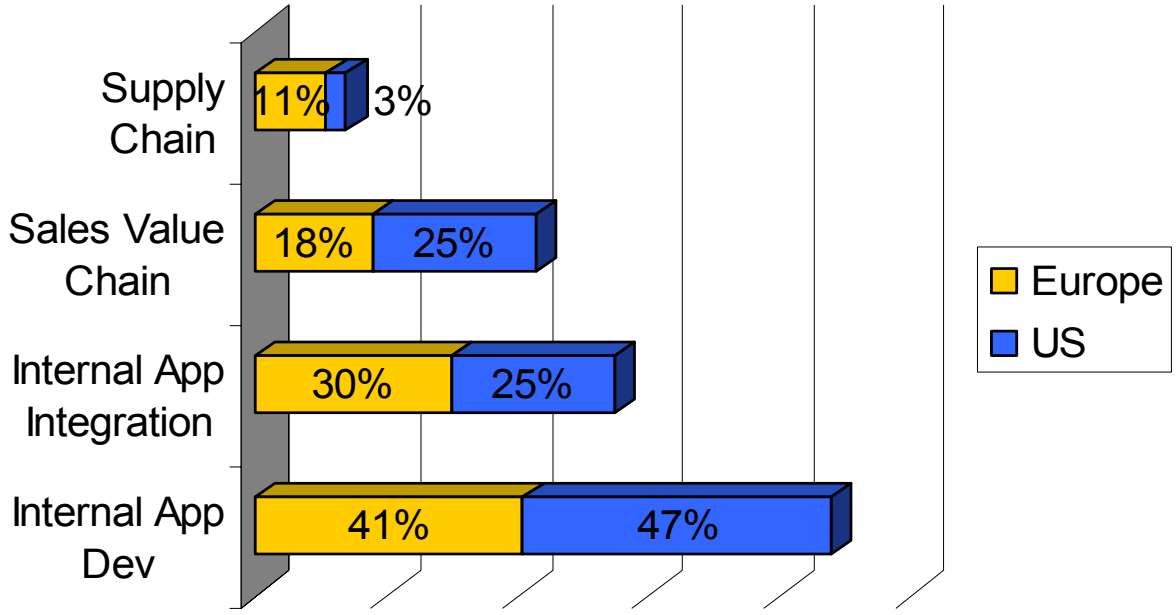
Technical Benefits



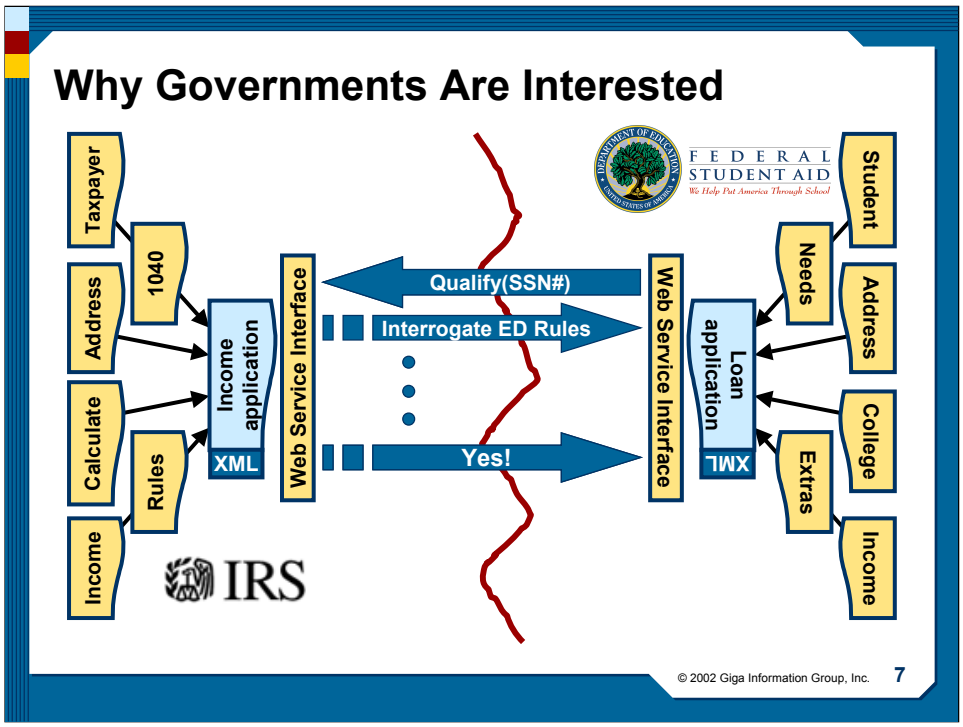
Source: Giga & GovTalk

© 2002 Giga Information Group, Inc.

Primary Target for Web Service Deployment



© 2002 Giga Information Group, Inc.



The example that was posed requires some fairly complex information flows. Now if we multiply this scenario over several times with other agencies, and also factor in the increased concerns about privacy, and security after 9/11, then the requirements for legal intervention by Congress become burdensome.

To some extent, Web Services may be able to mitigate the data security/data integration requirements if we assume that some departments like ED can surface their unique business rules (i.e., who qualifies for aid) as Web Services.

Then ED can ask the IRS (on behalf of the colleges and itself), by sending a Web Services message: "Does (SSN#) Qualify for Aid?. The information flow would then be:

ED ➔ Does SSN#123 qualify for student aid? ➔ to IRS

IRS ➔ interrogate ED rules, wrapped as services ➔ to ED

IRS <evaluate / calculate>

IRS ➔ Yes / no ➔ to ED

A response of Yes or No, mitigates the issue of ED needing to know income data (and the thorny issue of ED having to manage that info).

Business rules are being standardized in XML. Sending an XML representation of a business rule to be executed by the IRS in a DMZ, is an alternative (and more general approach to the above problem. Sending business rules as XML has the value of reducing the exchange/interrogation, is therefore more secure, and rules engines are becoming standardized. The new information flow with a rules approach would be:

ED ➔ Does SSN#123 qualify for student aid with **these** rules? ➔ to IRS

IRS <evaluate / calculate>

IRS ➔ Yes / no ➔ to ED

Web Services – Snapshot 11/02

- Still too many chiefs
  - No chief architect
  - No agreed upon definition
  - No agreed upon architecture
- W3C's role still uncertain
  - Patent dispute, Semantic Web distraction
  - Where does their role begin or end?
  - New Web Services activity
- Web Services Interoperability (WSI.org)
- What must you do?

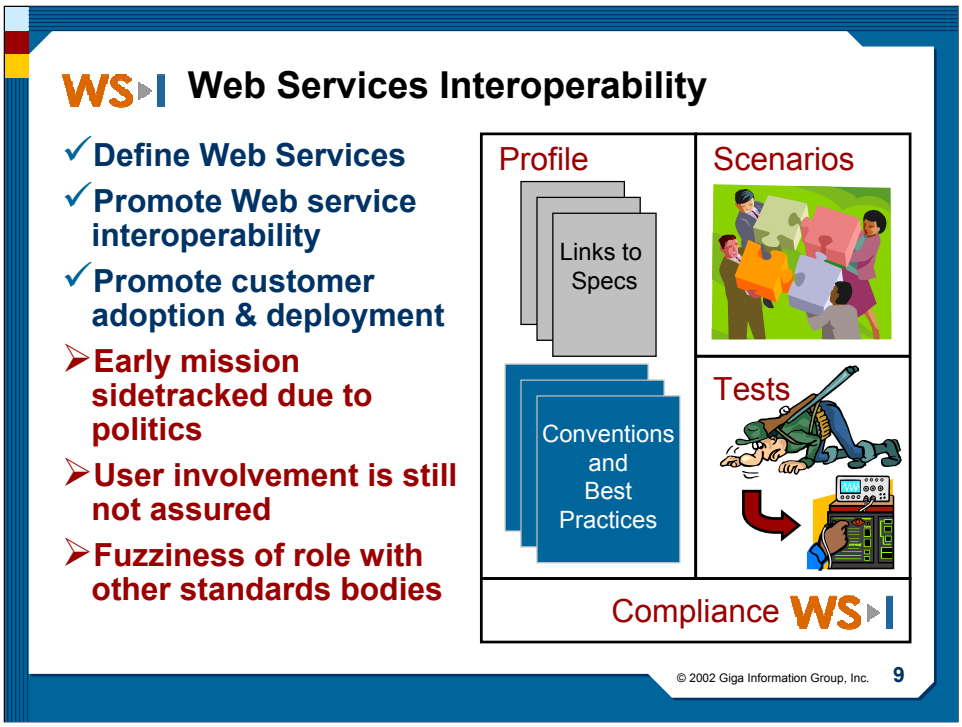
© 2002 Giga Information Group, Inc.

Last year, Web Services was threatening to collapse under the weight of incompatible vendor "additions." Today, there's a sense that, while it is still under construction, at least some direction is being defined. While partly true, there are still some issues that early adopters must confront.

There is no common, agreed-upon definition of what a Web Service is. Since there is no Web Services "architect," there is no agreed upon definition of a Web Services architecture. The Web Services vision is being driven by multiple, sometimes conflicting, visions of what Web Services are by vendors and users. This lack of a clear, common definition means that there are multiple logical Web Services architectures being proposed by vendors and users alike.

While there are still gaps where standards are needed, a more pressing problem is the increasing fragmentation and "niching" of Web Services standards, tools, APIs and technology. We have multiple security standards being proposed, multiple business process management standards, even multiple standards for reliable messaging at the wire level. While it's certainly true that many vendors put out standards as proposals, in order to stake a position, and then work behind the scenes to get agreement, the problem is that user involvement typically comes very late in the process. The key Web Services value proposition of interoperability is therefore jeopardized. The W3C, which in the past was the logical choice of arbiter. Saw that role diminished last year and is struggling to recapture it amidst a changed landscape featuring a new organization, WS-I.

For today, corporate architecture groups need to be more heavily involved than they normally would be in pilot and research projects, providing a clear Web Services vision, a common definition and a cohesive (if provisional) Web Services architecture.



WS-I plans to solve many of these concerns through three deliverables: profiles, scenarios and test suites:

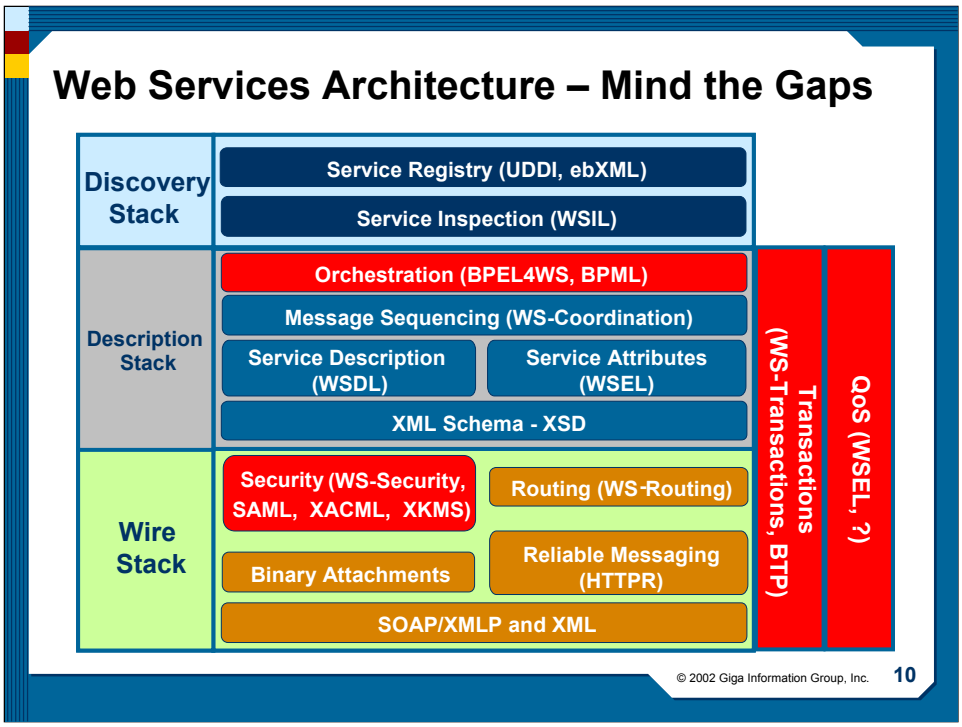
- Profiles group XML standards (and their versions), give guidelines/best practices on implementation, pinpoint how they work together and identify "gotchas." So, an example Web Services Basic Profile might consist of: XML 1.0, XML Namespaces 1.0, XML Schema Descriptions 1.0, SOAP 1.1, WSDL 1.1, UDDI 2.0. WS-I hopes that it will not need to create standards, merely ensuring interoperability between standards produced by other standards bodies (although it has not ruled out accepting any standards, even single-vendor, license-encumbered standards).

- Scenarios are focused examples, like interbank transfer, set within a given profile. Scenarios define specific Web services to be implemented. Each service is then used to exercise a specific set of functionality within a profile. WSI is hoping for significant input from the user community to help define these scenarios.

- The goal of the test suites is to allow a provider to self-certify that its Web Services conform to a named profile.

Much of the success of WS-I will be dependent on three things: the level of involvement of the user community; the internal structure of WSI; and the ability of WSI to liaise with and sometimes drive the creation of standards within other standards bodies.

Our recommendation is to get involved if possible; the quality of the results will depend on active user involvement, in part to preserve the vendor-neutrality, platform and language agnosticism and interoperability of Web Services. If you can't get involved, track the work closely. Monitor WS-I's progress and if stagnation sets in (the Web site not changing is one clue, and missed deadlines another), consider the effort dead and move on. Also, spread the word among the verticals that you participate in, so that scenarios that are representative of your interests are created for guidance.



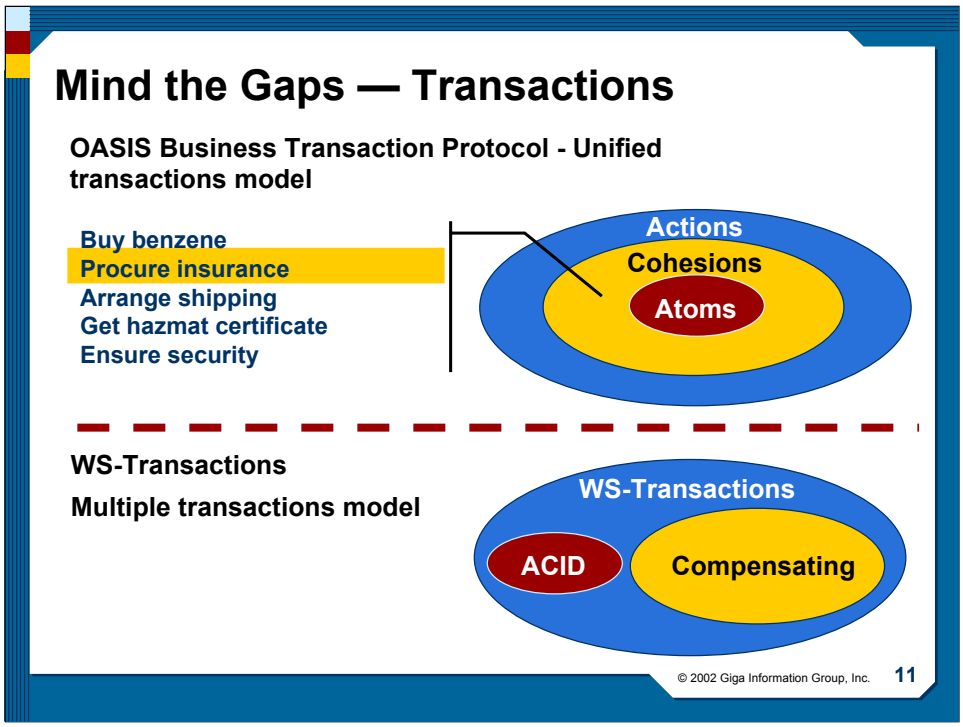
During the last year, the concept of a logical Web Services Architecture (WSA) has taken root. Proposals from IBM, Microsoft and several other vendors are converging on a common set of standards-based services that such an architecture must provide.

(Note that the presentation layer is not visualized here to preserve some simplicity! Proposals for the presentation layer are undergoing development with much work from OASIS.)

General consensus highlights the need for between 15 to 20 Web Services standards. Today, there is only widespread agreement on three: SOAP (and the W3C standardization of it, tentatively called XMLP), UDDI (backed only by an industry consortium, not by the W3C), and WSDL (which was submitted to the W3C, but has seen little activity in the last year). As you can see by the slide, there are several vendor proposals and proposals that are associated with other standards bodies like OASIS and IETF.

While consensus around a Web Services Architecture will take years to emerge, we urge you to create one, independent of any application architecture you have. Creating your own Web Services Architecture will be the best approach to balancing the immediate need for a platform-neutral integration layer, with the long-term need for managing its change and growth in alignment to business strategy.

While many of the “boxes” can be filled in today, the ones in RED (WHITE in a black and white printout) represent gaps that are critical to the success of Web Services to address all of the use cases discussed on slide 14). We discuss them in passing here, but they are covered in more detail on www.gigaweb.com.



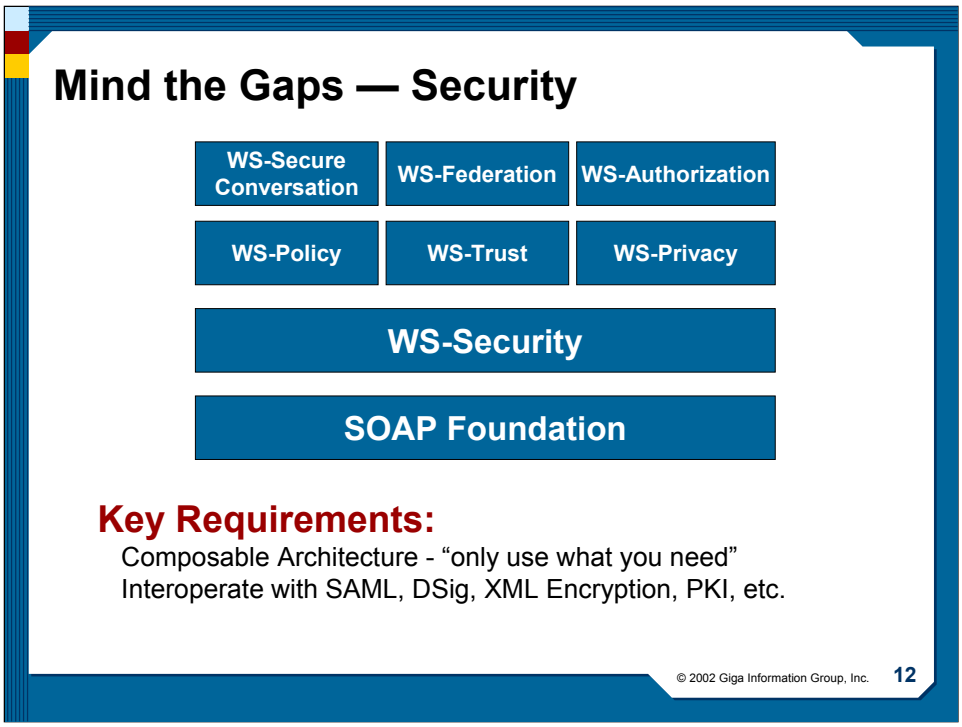
Almost everyone accepts that we need a standard that accommodates both classical ACID (XA or database-style transactions) and long-running, compensating transactions. But opinions are sharply divided on where such standards fit in the Web Services Stack.

The Business Transaction Protocol from OASIS has a number of smaller vendors (BEA, HP, Choreology, Oracle) on board and they released the 1.0 version of the spec in May 2002. BTP has been criticized by some as being too complex, and still lacks an industry heavyweight's backing, like IBM or Microsoft. So recently, IBM, Microsoft and BEA announced the specification for WS-Coordination– a general purpose coordination protocol, to be used with message sequencing and state machine synchronization and WS-Transactions, which includes support for both styles of transactions.

However, BTP tries to unify the two concepts, while WS-Transactions keeps almost two separate protocols in one. BTP defines the world of consisting of actions. Some actions have classical ACID (atomic, consistent, isolated and durable) properties. But others may have more complex semantics. Consider ordering a hazardous material like benzene, which requires arranging shipping, insurance, security, etc. Any failure should abort the purchase, but it's possible that if the business need is great enough, you may forego the insurance. So the sub-transaction, ArrangeInsurance, must be allowed to fail without failing the order. BTP calls this more complex transaction a cohesion, which can have relaxed isolation and possibly relaxed atomicity.

While in an ideal world, we could hope that the more elegant design of BTP may influence or change IBM and Microsoft's WS-Transactions, market realities will dub WS-Transactions the winner if it is royalty free and moves to an appropriate standards body. Our recommendations are to:

1. Review, comment, and help improve the specs when the process allows
2. Pressure Microsoft/IBM to ensure that the new specs are royalty-free
3. Engage in dialogue with IBM/Microsoft to ensure that the specs go to an appropriate standards body
4. Evaluate early implementations for inclusion in corporate architecture standards

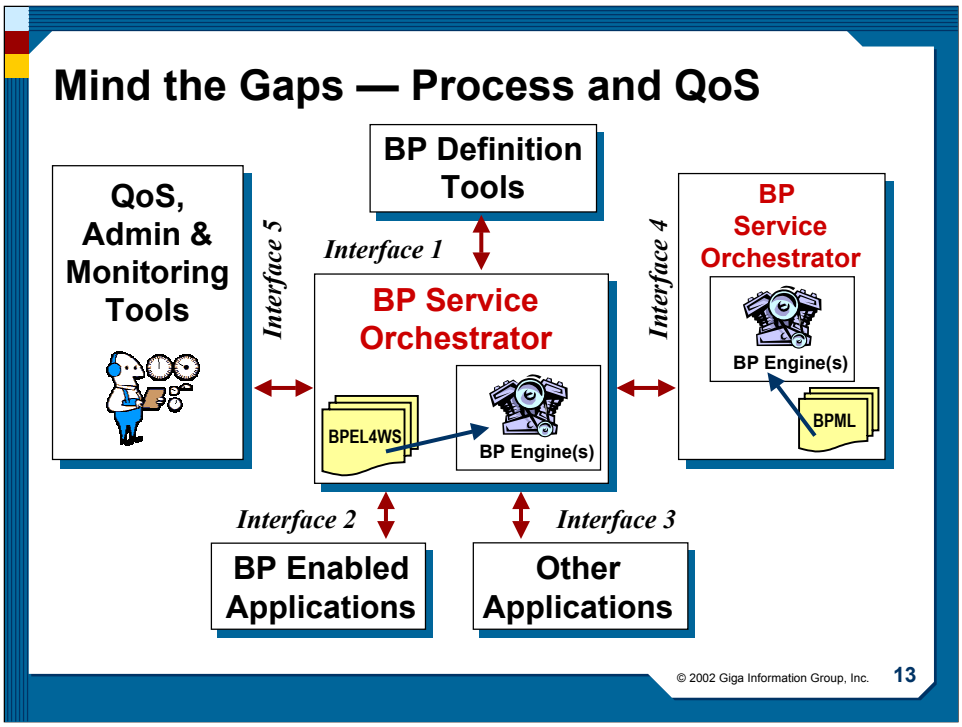


Giga’s surveys have consistently identified security as the number one limitation of current Web Services initiatives.

Yet, security is hard. The Internet was not designed with security in mind, since in the days of DARPA and ARPANet, everyone almost knew each other. So we can’t rely on any single management infrastructure, and, in general, security has to be end-to-end on the wire — we can’t be depending on something in the middle that “adds” security. Another requirement is that Web Services security must be compatible with foundational technologies such as SOAP, WSDL, XML Digital Signatures, XML Encryption and SSL/TLS.

Several initiatives are underway, including the new IBM-Microsoft set of specifications that were announced under the umbrella of WS-Security. The stack above depicts the road map of the recently announced WS-Security initiative from Microsoft and IBM. Their goal is to create a set of standards that allow corporate users to use only what they need, while depending on, and interoperating with core W3C security initiatives. A conservative estimate is 12 to 18 months to get the whole stack completed, but the base parts will be ready very soon. This is a valuable effort to fill the gap, and early adopters should get familiar with the early alpha/beta implementations. Note that these standards do *not* address cost of managing PKI certificates, which survey results have indicated is dampening adoption rates for resource constrained organizations. New technology, like the security appliance from Forum Systems, will help decrease some of the costs of managing XML-based security.

The migration of WS-Security to OASIS, and Sun joining the vendors supporting it, has solidified the industry behind it. We expect that by the end of 2003, that the core standards will have widespread agreement. Adoption will begin in earnest in 2003.

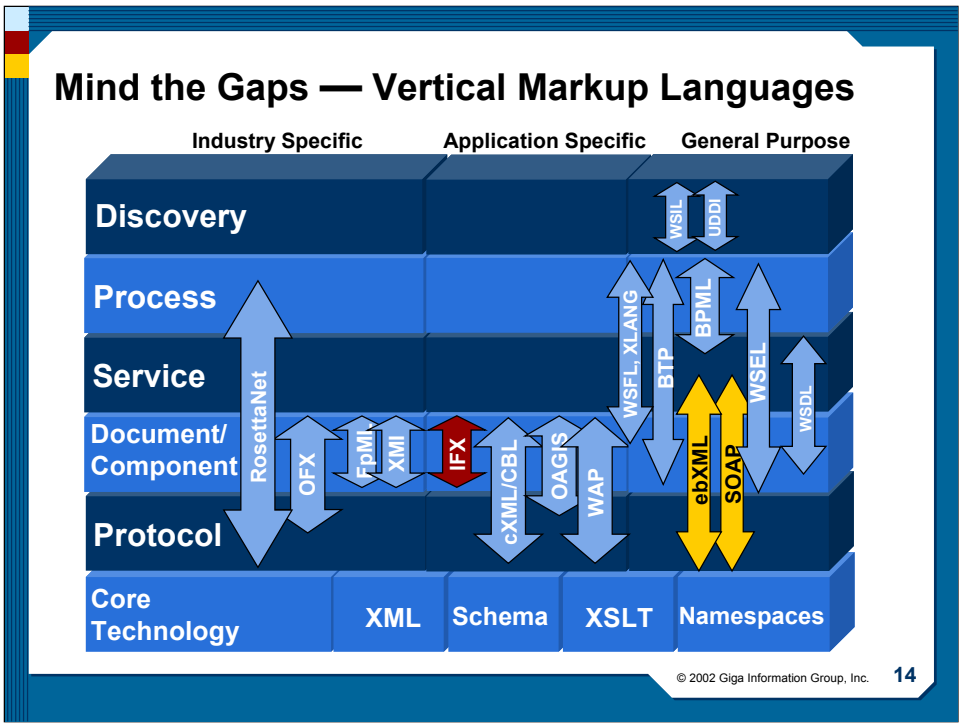


According to a recent Giga survey, quality of service (QoS) runs a close second to security in the minds of those considering Web Services. QoS and security together help determine trust. Current QoS initiatives are quite immature. For example, IBM hinted at a new QoS specification called Web Services Endpoint Language, WSEL, in the appendix to the WSDL spec, yet WSEL has not yet been released for review outside IBM. OASIS has some support for QoS, but only within the context of an ebXML message exchange. Microsoft has some support for QoS, but only within the context of a BizTalk message exchange.

What is clear is that QoS must be end to end, and also be relatable to the business process that the message exchanges are part of. Organizations have many business processes that are complex, oftentimes unique to a business, constantly evolving and involve both real time and long-lived transactions. But these processes are typically deeply embedded in applications which are difficult to centrally manage and integrate. A grand trend for the balance of this decade will be to get the process out of applications into an externally visible, editable and ultimately executable form. Current technology contenders are business rules systems and business process management systems.

The graphic above depicts an architecture for business process management (BPM) and workflow systems. Typically the BP Engine would operate on an XML representation of the process (candidates are: BPEL4WS, and BPML) and the architecture would allow for process description (design/redesign), deployment, execution, maintenance, optimization and analysis.

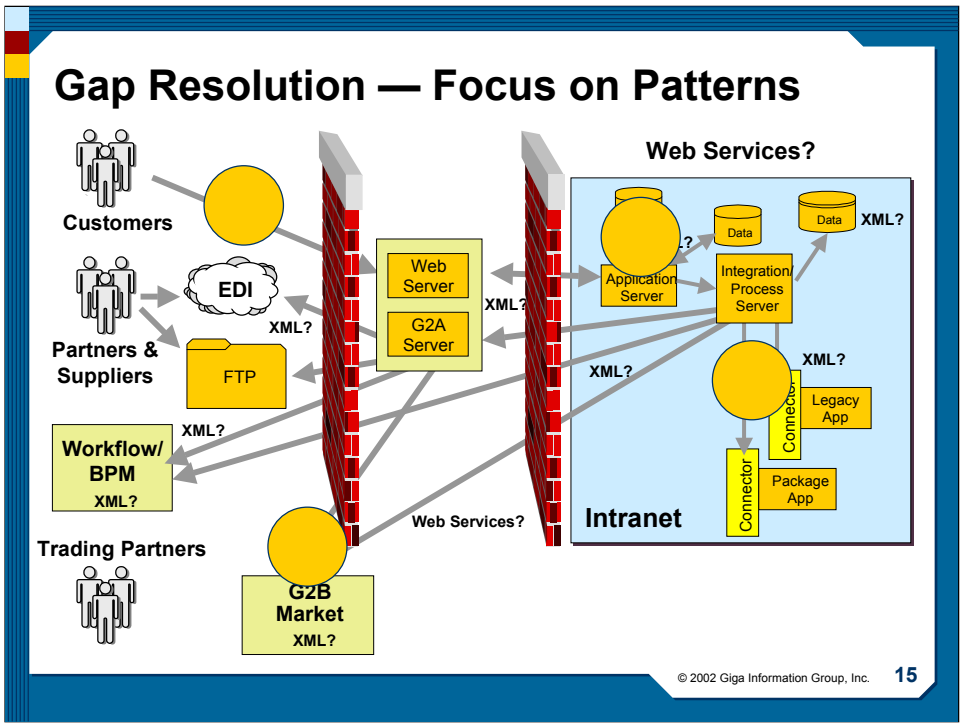
Web Services will be both an enabler and a consumer of the trend toward process management. The recent Microsoft/IBM/BEA announcement around their merging of XLANG and WSFL into the Business Process Execution Language For Web Services is a potentially market enabling development. While the resulting language needs work (resembling a strange mixture of the two base languages), keep an eye on BPM and plan for adoption of some BPM for managing Web Services in the 2004 timeframe.



The two dominant protocols for exchange of business data will be ebXML and Web Services (depicted as SOAP above). Currently, far too many document standards (like FIX) rely on their own protocols for routing and session management.

This makes it difficult to transmit vertical markup messages over horizontal transports like Web Services and ebXML.

We predict that *every* standard that relies on its *own* routing and session management protocol will drop it within two years, like OAGIS and more recently IFX did. But until they do, you will have to do the hard work of accommodating a vertical standard to Web Services or ebXML.



As one can see from the typical view of an n-tier application, the number of possible places where XML and Web Services can be constructively deployed is quite large.

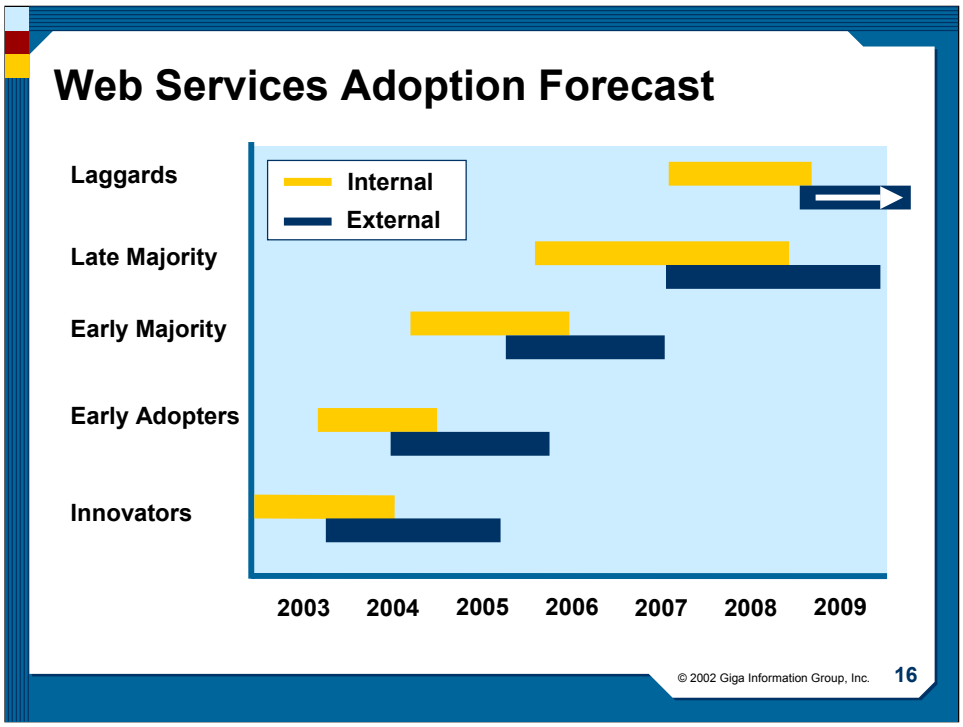
Never blindly add XML or Web Services everywhere. Deciding to where to use these technologies can be a complex decision because one has to balance functional and performance needs against the inherent flexibility that XML and Web Services adds. A functional need for evolving application configuration data, for example, doesn't require XML, however, using XML may allow one to evolve the configuration data format more gracefully.

The performance impact doesn't merely affect the network; larger messages mean more CPU time to parse and unparse messages, increased storage requirements (here the tag-to-text ratio is very important) and increased memory usage (especially if DOM-based parsing is used).

Similarly, a Web Services interface should emerge from business requirements, not from a decision to take an existing interface and "expose it" as a Web Service.

The key to accomplishing the above is through the use of patterns and guidelines, which will help manage the complexity and decrease the risk of adoption of XML and Web Services. Several industry patterns, such as Multichannel Interface, Spoke and Hub, Canonical Data, etc., are depicted above.

Focus on identifying patterns in the design and application strategy.



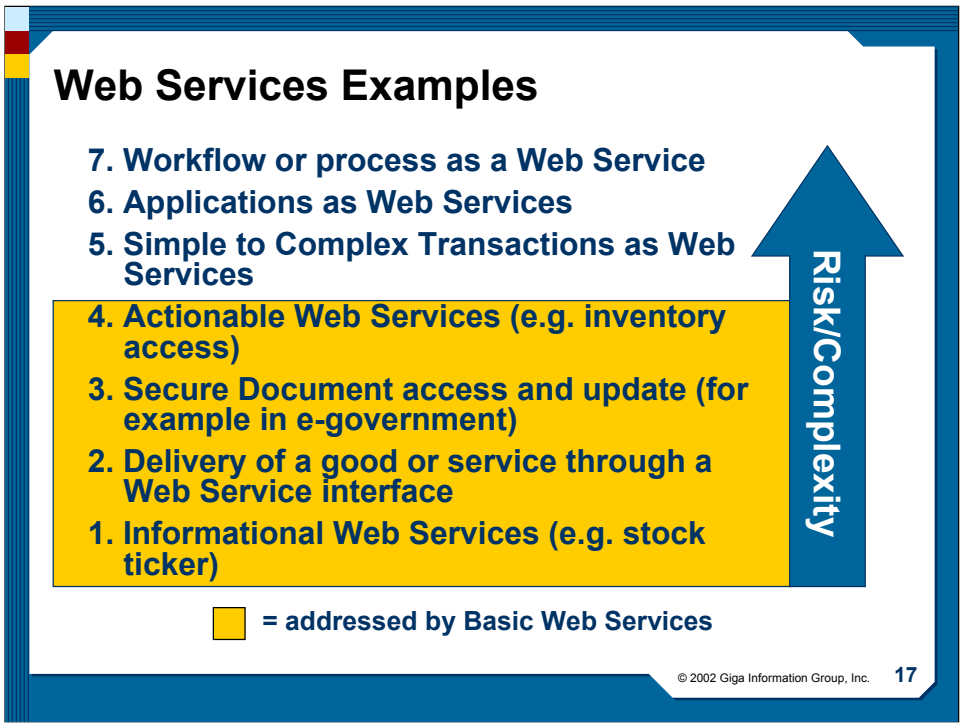
For some organizations, simple informational Web Services will be the first deployed, especially if the demands for security, and demands on the underlying infrastructure are light. For these companies, this will take the form of replacing a previous generation of Web-based technology (server-side scripting, CGI and the like). Others will have doubts that there is business value in replacing a technology (server-side scripting) that works.

For these organizations, creating access paths to stove-piped applications will be the compelling driver behind Web Services. Still others will see Web Services as a means to pursue an XML-EDI solution that will enable smaller suppliers to participate in a value chain. In other words, it is difficult to generalize about Web Services adoption patterns, since they will be largely driven by business need, vertical industry and the adoption profile of the organization toward technology.

We forecast that the most prevalent use of Web Services over the next 5 years will be as a standards-based approach to application integration — first internally, then with carefully chosen business partners. For the present, Web Services will not reduce the demand for integration servers as those servers clearly provide additional value in their support for mapping, routing, transformation, and publish/ subscribe, Web Services, however, will transform some aspects of the role these servers play.

Given these points, Giga predicts that the “Innovators” and “Early Adopters” of Web Services technology will develop and deploy Web Services for internal use in the 2002 to 2004 timeframe. In 2004, the “Early Majority” join the fray, and in 2005 the “Late Majority” followed by the “Laggards” several years hence. Because of the technology issues with security and messaging, external services follow internal service usage as depicted above.

Know who you are (early adopter vs. mainstream, etc). Consult what others in your vertical are doing. When to start preparing depends on the state of your IT environment and your time frame for implementation as indicated above. Since business demand for functional interoperability should be the driver for Web Services projects, the most likely candidate applications are those that the business has been eyeing for some time with the hopes of modernizing and integrating with newer applications. Yet IT-driven demand for Web Services may also be valid to drive small proof-of-concept (POC) projects that flatten the learning curve for future applications development. In well-funded IT organizations, these POC applications can begin immediately, while other organizations will wait until the economic climate improves.



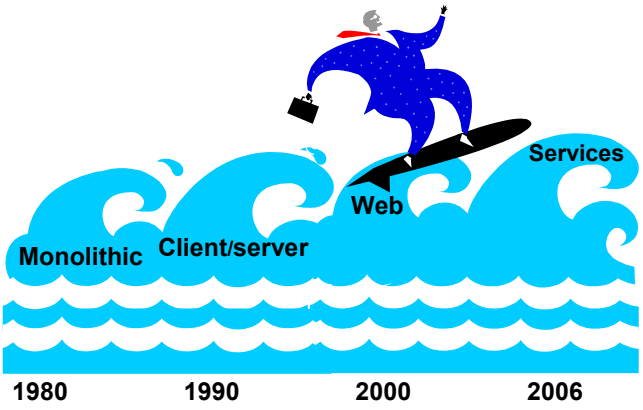
With the above limitations, it is no wonder that organizations are very cautious about where and when they first deploy Web Services. It seems reasonable to assume that adoption will follow the path taken by classic Web technology, along a continuum from low-value, non mission-critical, to high-value, mission-critical. The slide depicts examples of Web Services, from lower complexity and lower risk, to higher complexity and higher risk. It's worth noting that these examples can potentially be either internally or externally deployed Web Services; for the case where it's external, the need to provide security raises the risk and complexity level over the internal form.

Service-based design identifies and rationalizes candidate services which must be identified and rationalized with the business objectives. Not every function will be a service, nor should they be. In theory, a service boundary should align with commonly needed functions, represent a complete unit-of-work, and have a clear and communicable end result. In order to be good candidates for reuse, the functions should be in fairly constant demand. Examples of simple services may return a customer balance, credit-status or order-status — each are valuable in many applications and return a fairly discrete answer. Reporting functions can also be good early examples of services that provide business value, yet carry little risk.

With candidate services identified, examine the underlying technology of those applications — are they stable and mature, is it possible to wrap the underlying application functionality as components so that services can be layered cleanly on top? How do the standards tap the existing systems — do the applications have a viable API, can they be accessed via the presentation layer using Web-to-host tools, or is application mining a viable approach? How large a factor is transaction latency? A tactical adoption approach minimizes organizational and process change in favor of quickly adopting Web Services for high-value business needs. A strategic approach will focus more on the transformation of the organization and process, to accommodate a services-orientation — with each line of business analogous to a service bureau. These service bureaus identify their business "plug points" — where they naturally interact to exchange discrete information. The plug points and their relationships equate to service interfaces that will enable the business to reconfigure itself. Adopting a services-orientation requires you to identify the cohesive business interfaces that constitute a service as well as the loosely coupled relationships between the service units. Successfully adopting a strategic approach will require a strong architectural focus.

Recommendations

- **Be *selfish*. Be *pragmatic*.**
  - Keep what works (legacy)
  - Focus on business needs
  - Let your vertical (external XML standards) guide you
  - Create your own Web Services architecture
  - Mitigate Risk with WS-I
  - ✓ Perform impact analysis on Web Services' effect on bandwidth and CPU utilization



- **Be *opportunistic*.**
  - Brainstorm new business opportunities
  - Re-evaluate business models for new partnerships
  - Proactively manage/retire what won't transition
  - Adopt a service oriented architecture for new development

© 2002 Giga Information Group, Inc.

Set the hype aside and evaluate Web Services for what they are today — a new approach to connecting applications that holds great promise, but contains certain aspects that remain immature for some current projects. Be wary of using the external form of Web Services and focus on whether, where and when your organization should adopt Web Services as an approach to internal application integration.

Begin planning for the adoption of XML and Web Services by identifying high-value examples that fit the technology-adoption profile of your company. Innovators and early adopters of technology can act today, the trailing minority and majority should begin to consider pilot projects to prove / disprove the concept and its feasibility for adoption with your planning horizon. Focus on the WS-I's Web Services Basics profile as the base upon which you build first-generation Web Services.

The security, QoS and reliable messaging gaps make widespread exposure of meaningful transactions to the public internet impractical at this time. Focus on how Web Services can simplify the integration problems internally until such time as these issues are resolved — not likely before Q4 2003 at the earliest. Consider whether the document XML approach will solve latency issues. Begin analysis of the WS-Security proposals to see what will be usable within your planning horizon.

Educate the organization's business leaders on the current and future capabilities of Web Services. Let educated business need drive which services get the attention of Web Services developers. Draw candidates for "services" from the highest-demand integration projects. Applications that were candidates for Web-to-host tools — the existing functionality is valuable enough to keep, but requires browser or multi-channel, multi-modal access — are likely to be excellent candidates for Web Services. In applications with no discernable API, Web-to-host tools provide a number of options to expose business functions to the Web Services standards, and in that context, enable Web Services to operate in conjunction with custom-built legacy applications.

We are in the midst of a rebuilding of EAI technology on top of a Web services foundation and efforts to create standards to ensure that such a Web services based EAI will remain broadly interoperable across platforms. Yes, integration is hard work; Web services don't alter that basic fact, which means we foresee a slow but steady adoption of Web services. But the value proposition of this new Web services integration, or WS-EAI, to honor classical EAI, will be fundamentally different, since it brings to the forefront the term "services, " a weak concept in present day EAI. Service-orientation allows Web services to offer a longer-term vision whereby business functions within and across enterprises interoperate without regard to their technological underpinnings. As such, they will, when mature, provide a significant step forward for integration.