Patricia Seybold Group

Strategic Consultants & Thought Leaders

Patricia Seybold Group's Executive Series

An Executive's Guide to Web Services

How to Optimize Web Services Investments to Improve Your Customer Experience

By the Patricia Seybold Group

85 Devonshire Street, 5th Floor, Boston, MA 02109 • Phone 617.742.5200 • Fax 617.742.1028 • www.psgroup.com

TABLE OF CONTENTS

What's Next? If you find this report valuable, then:
Print It – YES, you are free to print and freely distribute this report as long as its contents are not changed.
Send It – Send a friend or colleague to http://www.psgroup.com/vm/ws/ so they can download their own copy.
Stay Informed – Subscribe to our free e-mail newsletter at http://www.psgroup.com/signup.asp to stay up-to-date on this and other important research topics.
Contact Us – Contact us at feedback@psgroup.com to find out how our additional research and consulting services can help your organization sort through its Web Services strategy.

Web Services Guide for Customer-Centric Executives

How to Optimize Web Services Investments to Improve Your Customer Experience

By Patricia B. Seybold *May 2002*

FOREWORD

Why are customer-centric business executives all around the world paying attention to the latest "buzz term" in the IT industry—Web Services? They recognize that Web Services' standards suddenly make it easier for them to meet business customers' and channel partners' demands. Web Services make it much simpler to integrate disparate applications both within and across organizational boundaries.

The common wisdom you'll hear from most analyst firms is that Web Services' architectures will be implemented initially *within* companies. Our competitors believe that Web Services will be cautiously piloted as an approach to *internal* enterprise application integration (EAI). Yet, we've discovered that many of the earliest production implementations of Web Services are being used to support business customers and trading partners. So, instead of focusing first on internal application integration, forward-thinking companies are starting from the *outside in*—leading with customer-connectivity.

Take, for example, Cisco Systems' product configuration application. Since the mid-90s, Cisco has offered a Web-based configurator on its public Web site. And, for its largest business customers like AT&T, SBC, and others, Cisco had to install and maintain a custom-configured version of that application on a server located within each major customer's firewall. In early 2001, Cisco re-developed that product configurator as a Web Service. Now Cisco maintains a single Web-based application, which presents itself as a Web Service poking through each major account's firewall, integrated into that account's own procurement application. Here's another example: General Motors uses Web Services to make many of its operational applications available to its dealers' personnel by integrating key functions, like credit approval, into the dealerships' own applications.

So, here's your primer—everything that you, as a business executive, need to know to get started with your own Web Services strategy.

Web Services Defined

Here's our description: A Web Service is a URL-addressable software resource that performs functions and provides answers. A Web Service is made by taking a set of software functionality and wrapping it up so that the services it performs are visible and accessible to other software applications. Web Services can request services from other Web Services, and they can expect to receive the results or responses from those requests. Web Services may interoperate in a loosely-coupled manner; they can request services across

the 'Net and wait for a response. A Web Service can be discovered and leveraged by other Web Services, applications, clients, or agents. Web Services may be combined to create new services. And they may be recombined, swapped or substituted, or replaced at runtime.

You'll get an easy-to-assimilate picture of how Web Services actually work in the enclosed chapter entitled, "Why Web Services Are Strategic," beginning on page 12.

The 9 Questions You Should Be Asking

Even before you delve into the details of Web Services, we recommend that you begin by asking yourself one simple question: *What would my Customers (or partners, or suppliers) like to be able to do that's currently difficult for them to do?* In other words, what kinds of information do they need that's currently difficult for them to access? What activities would they like to be able to undertake from within their own company's applications that would involve connections to your company's business systems? Maybe your customers or channel partners need visibility into your inventory before they can place orders? Perhaps they need to be able to monitor their own transactions. Maybe they need access to particular applications you have available internally or on your Web site? Make a list.

Once you've come up with a list of functions and capabilities that your customers and trading partners need from your firm, pass that list along to your IT executives along with our 9 questions in the Report entitled, "How to Think About Web Services," starting on page 4. The answers to these 9 questions should form the blueprint for your company's Web Services' priorities and architecture.

What's the Adoption Time Frame for My Company?

How soon should you be prepared to roll out Web Services and for which types of applications? That will depend on your company's culture. Is your organization cautious when it comes to adopting new technologies? Or is your company aggressive in jumping on the latest IT trends? Perhaps your organization falls somewhere in between? We offer some guidelines for you to use in gauging the rate at which you should plan to roll out Web Services for different types of applications in the Report entitled, "The Web Services Freight Train," on page 8.

How to Use Web Services to Address Pressing Business Requirements

Once you've pinpointed the areas of functionality you should be addressing by answering the 9 key questions, and you've decided what implementation speed is appropriate, it's time to prioritize. Now it's time to read (or re-read) the section entitled "How to Use Web Services to Address Business Requirements" starting on page 19 in the Report, "Why Web Services Are Strategic," starting on page 12. Notice that we make two important recommendations:

1. Use Customers' Key Scenarios to help you prioritize which Web Services to do in which order. You want to focus first on those Web Services that will support the greatest number of customer scenarios across customer segments.

2. Identify and standardize on a common set of Web Services' infrastructure services, such as directory services and logging and reporting services. You'll find our list of candidate infrastructure services on page 25.

What Are the Key Categories and Selection Criteria?

As a business executive, you won't need to concern yourself with the detailed technical trade-offs required in evaluating the Web Services solutions that your firm selects. However, we strongly recommend that you inform yourself about the different categories of tools involved. As you'll discover by reading "Anatomy of Web Services" on page 29, a casual choice of tools in one inexpensive category, such as Web Services development tools, may lock you into a choice you might not have made when it comes to the more expensive and more strategic decision: your firm's choice of Web Services deployment platform and infrastructure services. What you don't want to do is to foster an inadvertent proliferation of deployment platforms, nor of core infrastructure services.

What to Look for in a Web Services Assembly Environment?

A good place to begin your investments and your experimentation is probably in the relatively low-risk area of development and assembly environments. But you'll still need to select a toolset that's geared to the software environment into which you're planning to deploy. In today's market, that means you're going to need to decide whether you're most likely to want to deploy first on J2EE or .Net environments. You'll find some useful guidelines in the Report entitled, "What to Look for in a Web Services Assembly Platform," on page 40.

Next Steps?

I believe that this *Web Services Executive Guide* will provide much of the background and context you'll need to take advantage of the Web Services wave. We don't think this is just another over-hyped "marketecture." We feel that this Web Services wave is part of an inevitable IT architectural evolution towards Service-Oriented Architectures that will give business more flexibility in responding to and in anticipating customers' constantly changing needs. We know that many firms are piloting the use of Web Services for internal application integration. While this is not a bad idea, we think that if you adopt this approach as your Web Services strategy, you're going to be left breathing the exhaust of more forward-thinking competitors.

We encourage you to distribute this *Web Services Executive Guide* to your colleagues (see the "What's Next?" table below the table of contents and on the last page of this Report). We believe that it will help you establish a common language and lens through which you can study the turbulent Web Services horizon.

As you sort your way through your own Web Services strategy, we'd be happy to help. We offer lots of additional research in our Strategic Research Service.

We can also advise you on a consulting basis—to provide an objective expert opinion on where you may be missing the boat and/or to help your own business and IT team with Web Services strategy, prioritization, and platform selection.

And we offer a customer-centric methodology, Quality of Customer Experience (QCE)[SM], that will enable you to transform your business and to achieve maximum customer profitability. It comes with tools you can use—Customer Scenario[SM] Design with Web Services identification and prioritization add-ons.

An Executive's Guide to Web Services

How to Think About Web Services

Questions to Ask in Planning Your Web Services Strategy

By Patricia B. Seybold

START FROM THE OUTSIDE IN!

The common wisdom circulating about Web Services is that the "safe way" to experiment with them is to pilot the use of Web Services for internal application integration. In short, simply use XML and SOAP as the way to do your internal applications integration—a next generation EAI. While this is not a bad idea, we think that if you adopt this approach as your Web Services strategy, you're going to be left breathing the exhaust of more forward-thinking competitors.

What we see happening among our visionary customer-centric clients is that they actually are starting from the outside in. They use Web Services as a straightforward way to encapsulate functionality from robust, production applications in order to expose that needed functionality to their customers, trading partners, and distribution partners.

Every company has a set of production applications. These are the ones that run your business. This is the first place to look for candidate Web Services that can be re-used and re-combined. What's nice about these production applications is that they are already robust, secure, operational functions. In other words, security, transaction integrity, and performance are already proven. These are trusted applications.

Don't think of an entire legacy system as a single Web Service. Instead, think of the various functions it performs, e.g., order tracking, inventory management, credit card processing, currency conversion, account reconciliation, production scheduling, and so on. Each of these capabilities is a candidate Web Service.

By encapsulating production functionality into a Web Service, you can make it more readily available to be re-used by other internal and external applications. You can unbundle a service from an existing application and re-use it to support a number of applications.

So, start your Web Services strategy by identifying the core capabilities that your customers and partners need today. There are a host of them: order status, delivery status, order placement, inventory availability, pricing, service activation, billing inquiry, and so on. The good news is that you've probably already exposed many of these services on your Web site(s) or portals. Now you need to re-architect those Web-access mechanisms to take advantage of Web Services. As you do so, you'll gain a triple whammy. You'll have more robust portal/Web architecture. You'll be able to offer this functionality as Web Services to customers, partners, and suppliers. And this will save you time and money doing one-off integrations with your external stakeholders.

We believe that you should use a Web Services approach to give you rapid time-to-market in connecting your customers, business partners, and suppliers to your core operational functions.

QUESTIONS YOU SHOULD BE ASKING TO JUMPSTART YOUR WEB SERVICES STRATEGY

1. What would my Customers (or partners, or suppliers) like to be able to do that's currently difficult for them to do?

2. Could we provide that capability using a Web Service?

3. What application functionality would we need to wrap into a Web Service?

4. Is that functionality already trusted, robust, and secure?

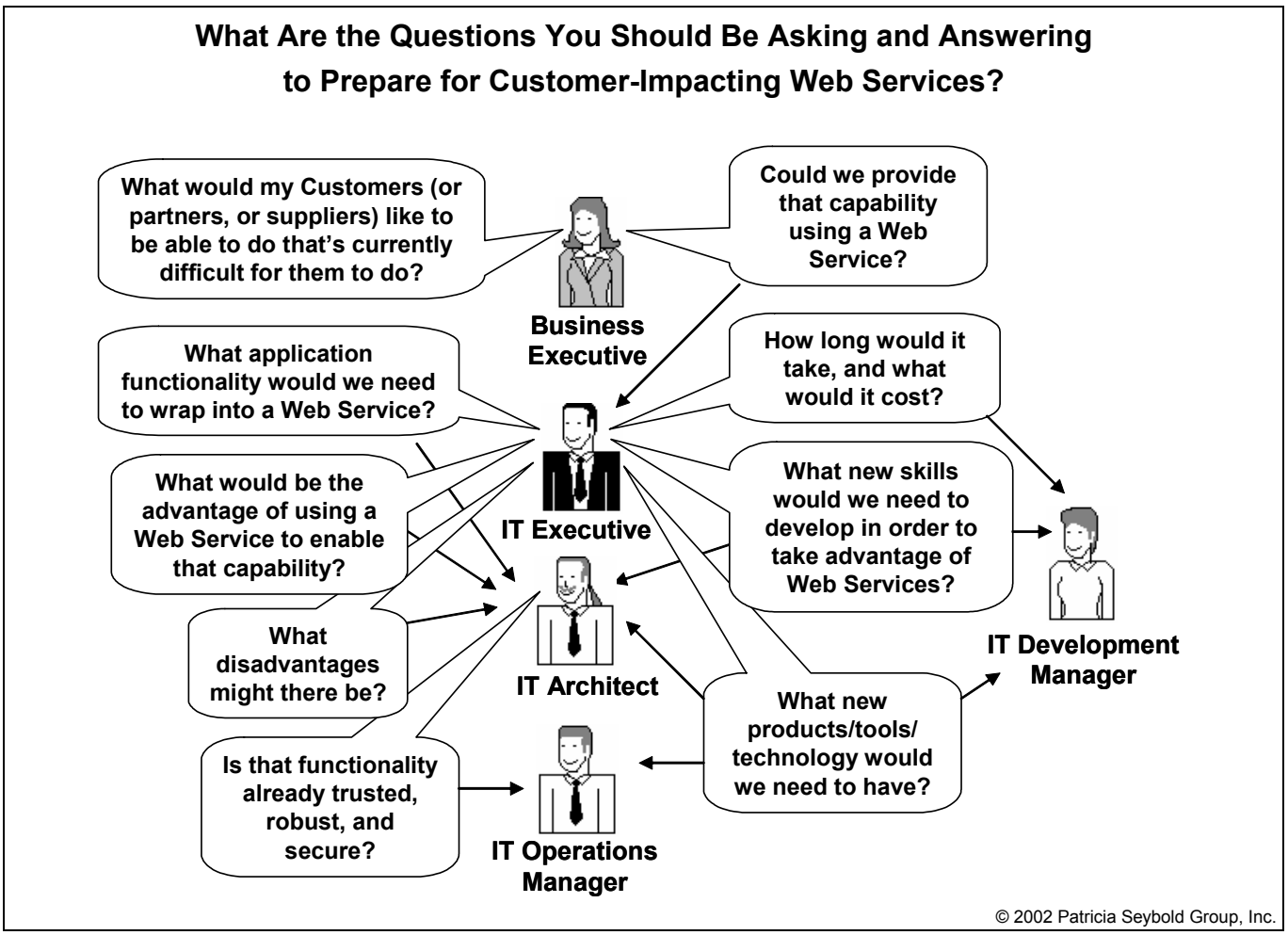

Illustration. We recommend that you begin your Web Services planning by focusing on what services/functions your large customers, key channel partners, and trusted suppliers need to be able to access from their own applications. Then make sure that your team has good answers to all of these questions. Don't overlook the need to have agreed upon security policies in place on both sides.

5. What would be the advantage of using a Web Service to enable that capability?

6. What disadvantages might there be?

7. How long would it take, and what would it cost?

8. What new skills would we need to develop in order to take advantage of Web Services?

9. What new products/tools/technology would we need to have?

If you're a business executive, sit down with a piece of paper and write down all the answers that come to mind for the first question. Gather input from your peers, from your customer support folks, from your e-business folks, and from your sales executives, partner management executives, and supply chain executives.

Then, meet with your IT architects and ask them to answer the following nine questions. (They may also have some insightful recommendations for the first question that didn't occur to anyone else.)

This is the most strategic way for you to be thinking about Web Services.

Patricia Seybold Group

What Are the Questions You Should Be Asking and Answering to Prepare for Customer-Impacting Web Services?

Business Executive	
What would my Customers (or partners/ suppliers) like to be able to do that's currently difficult for them to do? List the capabilities (e.g. place an order, check inventory availability, access transaction history) and/or the Information (e.g. access current prices, review most recent product specs, check delivery status, etc) that your key external stakeholders would most like to be able to access from within their own company's applications and/or from within their Intranets (without logging on to your Web site).	Customers: Partners: Suppliers: Others:
Which of those capabilities could we provide using Web Services?	
IT Architect	
What application functionality would we need to wrap into Web Services in order to provide these capabilities?	

What Are the Questions You Should Be Asking and Answering to Prepare for Customer-Impacting Web Services? *(continued)*	
IT Architect *(continued)*	
What would be the advantage of using a Web Service to enable that capability?	
What disadvantages might there be?	
IT Operations Manager	
Is the functionality we plan to expose as a Web Service already trusted, robust, and secure?	
IT Development Manager	
How long would it take, and what would it cost to encapsulate or develop each Web Service?	
What new skills would we need to develop in order to take advantage of Web Services?	
What new products/tools/technology would we need to have?	

Table A. Pass this worksheet around to build consensus about which Web Services may be the most promising ones for your organization to provide to key stakeholders. (You may want to fill in a separate worksheet for each Web Service.)

An Executive's Guide to Web Services

The Web Services Freight Train

Rapid Adoption Is Certain; Get Rolling or Get Rolled Over

By Susan E. Aldrich

NETTING IT OUT

Web Services will exhibit a steeper adoption curve, and infiltrate more of the applications portfolio, than previous application technologies. Web Services are in volume use today by technology-aggressive corporations; technology-moderates will be deploying production Web Services this year. Even the technology-cautious will be investing seriously by 2004. By 2005, Web Services will be present in the majority of business applications for both aggressives and moderates, and the cautious will deploy Web Services in the majority of strategic applications by 2006.

Web Services will not replace existing applications or middleware, but critical application functions will be wrapped and exposed as Web Services in order to support new streamlined and adaptive business processes.

There are two fundamental reasons that adoption will occur, and occur fairly rapidly. First, Web Services solve some expensive and intractable business and IT problems. Second, even corporations cautious about adopting new technology will be driven by customers and suppliers to adopt Web Services sooner rather than later.

Business and IT executives, and IT architects, should be scoping, valuing, prioritizing, and planning their companies' Web Services moves now.

BUSINESS CONTEXT

Web Services are on a swift adoption curve because they solve expensive and intractable business and technology problems.

The business problems are interrelated and reflect every company's current reality: stove-piped but efficient processing systems. Here is a short list:

- Companies need to respond quickly and reliably to special requests from customers and to special offers from suppliers.
- Business processes need customized tweaks for various situations, customers, suppliers, and changes in the market place.
- Business processes need to interact automatically, across organization and enterprise boundaries.

The technology challenge lies in creating adaptive business processes from stove-piped, incompatible, and isolated systems. In order to do that effectively, development teams need a way to "meet in the middle" without constant communications about the details of their interfaces.

For these reasons, companies will embrace and pervasively deploy Web Services. Adoption is driven (and inhibited) by technology maturity, corporate attitudes toward technology, business relationships, and the architectures of existing applications.

Technology Maturity

Web Services standards and technologies are mature enough to support three phases of Web Services applications:

Adoption Time Frames, Phases, and Infiltration

Time Frame for First Deployments	Corporate Technology Attitude	Web Services Phase	Likely First Applications	Application Portfolio Infiltration
Now	All	Exploratory	Any	None
Now	Aggressive	Preparatory (internal)	Business Process Integration	Majority of business applications by 2004
Now		Controlled (external)	Order Status Query Order Entry Purchase Order	
8 months	Moderate	Preparatory (internal)	Order Status Query Order Entry Business Process Integration	Majority of business applications by 2005
14 months		Controlled (external)	Order Status Query	
2004	Cautious	Preparatory (internal)	Order Status Query Order Entry Purchase Order	Majority of strategic business applications by 2006
2005		Controlled (external)	Order Entry Purchase Order	

Table A. This table depicts the time frames in which Web Services will be deployed by companies with varying attitudes toward technology, the phases of those deployments, examples of types of applications, and the year in which Web Services will be part of most applications.

- **Exploratory.** In the *exploratory* phase, IT is learning to build and to deploy Web Services, to design effective Web Service application architectures, and to control and manage a Web Services application environment. Virtually all companies employing programmers have someone exploring Web Services today.

- **Preparatory/Internal.** In the *preparatory* phase, IT develops and internally deploys Web Services that will, in the future, interact with applications outside the line of business (LOB) or enterprise.

- **Controlled/External.** In the *controlled* phase, these services are offered to external users, those outside IT's span of control. If these applications have stringent requirements for transaction integrity, the Web Services providers and consumers will negotiate how each will handle long-running transactions, exceptions, authentication, and message integrity, for which no Web Service standards yet exist.

STANDARDIZED. By 2005, Web Services standards and technologies will mature sufficiently to support the *standardized* phase of applications. In this phase, trading partners don't have to be in a close relationship and establish how to manage business process integrity in order to provide and consume Web Services. In order for standardization to take hold, three key transformations must occur. First, the missing standards must emerge. Second, the roles of development, deployment, orchestration products have to be established. Finally, patterns for basic interactions in an industry or community have to evolve.

Corporate Attitude toward Technology

It's useful to look at three corporate technology attitudes: the technology-aggressive, the technology-moderate, and the technology-cautious. Technology aggressive corporations always study emerging

technologies and invest early in the most promising. The aggressives have already deployed Web Services on a large scale, both for internal integration and external interactions. For example, Deutsche Telekom has five million mobile subscribers whose content is provided courtesy of Web Services.

Technology-moderates tolerate technology that has matured past the proof-of-concept phase and shows potential for improving costs or revenues. The moderates will be deploying preparatory Web Services applications within eight months, and exposing them outside in a controlled environment six months later.

Cautious companies don't invest in new technology until it is proven competent, valuable, and widely-adopted. The cautious will succumb to pressure, deploying Web Services by 2004 and exposing them externally by 2005. (See Table A.)

Web Services are more likely to use existing application software than to replace it.

Business Relationships

Since technology attitudes are not uniform among trading partners, IT's caution regarding Web Services will be overwhelmed by the demands of technology-aggressive trading partners. It's certain that, in the next two years, all companies will hear customers and suppliers expressing strong preference for transacting via Web Services. Within four years, companies unprepared for Web Services will experience revenue loss or supplier price increases.

Architectures of Application Portfolio

The expense and speed with which Web Services can be deployed depends on the architectures of the application portfolio and the architecture models of corporate developers. Web Services are more likely to use existing application software than to replace it. Applications can quickly be deployed as Web Services if they are already architected for loosely-coupled, asynchronous interaction, such as applications currently driven via a Web interface and used by trading partners. In contrast, application architectures that are monolithic, procedural, tightly-coupled, and demand synchronous interaction are poorly suited to deployment as Web Services. They will require at least moderate, and perhaps extensive, investment to provide Web Services that deliver the sought-after benefits of streamlined and flexible business processing.

The people who will develop Web Services will apply the architectural and component archetypes that are familiar. If today they work on tightly coupled and synchronous applications, they may have difficulty developing for the new service-oriented-architecture (SOA) paradigm.

Web Services Game Plan

Web Services will provide highly-effective application and process integration for both inter- and intra-enterprise interactions. External integration will be the killer application for most companies, easing the pain of value-chain integration.

For this reason, it is imperative that business and IT executives meet with key suppliers and customers to understand their business process issues and to share Web Services plans. By inspecting ways to optimize existing processes (and policies) to eliminate exceptions, ease bottlenecks, and enable custom handling, trading partners can identify the priority investments in Web Services.

Some of these business processes will have features not yet supported by Web Services standards. Trading partners can negotiate the details of how these features will be provided and establish time frames for implementation.

Business executives must set goals for inter-organization interaction, based on what set of interactions customers and suppliers will expect in three years.

IT executives must refine technology strategies for inter-organization interactions, identifying what services should be shared across divisions, and which application functions need to be shifted to a SOA.

IT architects must establish a Web Services architecture, with a primary goal of separating common and application infrastructure services from application logic.

CONCLUSION

Web Services will be pervasive by 2005, infiltrating the majority of strategic applications. This is not to say that Web Services will replace other middleware, and certainly existing applications will not be rewritten in Web Services. But, as new, streamlined business processes emerge, they will require various functions in the application portfolio be delivered on demand. These on-demand functions will be delivered by wrapping existing application steps as Web Services.

Business and IT executives should begin now to set goals, priorities, and plans to prepare for the impending pressure from customers and suppliers to use Web Services.

An Executive's Guide to Web Services

Why Web Services Are Strategic

A Web Services Primer for the Customer-Centric Executive

By Patricia B. Seybold

NETTING IT OUT

Why are Web Services more than a passing fad? Why should you make a Web Services strategy part of your business strategy?

The Web Services wave is a current instantiation of an inevitable IT architectural evolution towards service-oriented architectures. Service-oriented software architectures will give your business more flexibility in responding to and in anticipating customers' constantly changing needs. They also make it much easier to integrate disparate applications both within and across organizational boundaries. Web Services are the first easy-to-implement and easy-to-understand incarnation of a services-oriented architecture.

From a technology strategy standpoint, you can use Web Services today to prioritize and to rationalize your applications architecture and software infrastructures. If you do it right, you'll gain increased flexibility and more dynamic adaptability with lower implementation time and costs.

Start from the outside in! From a business strategy standpoint, you should use Web Services first to expose the functionality of your core operational applications to customers, business partners, and suppliers. Next, you should focus on building flexibility into your end-to-end business processes. Then, you should begin to think about how to leverage the software services that represent the core knowledge of your business—Web Services you'll want to deploy over the next three years.

What are the downsides? Although it's easy to create Web Services, very few software developers know how to design loosely-coupled architectures. They are used to "hard-wiring" applications together. You'll want to establish a core team of seasoned architects to lead your Web Services strategy and to educate your internal and external developers.

Here are the questions we'll answer in this Report:

1. How will Web Services help make your business more adaptive?
2. What are Web Services, and what do you need to understand about their characteristics?
3. How can Web Services bridge the communication gap between business execs and technologists?
4. How can you use Web Services to address your business requirements?
5. What are the pitfalls you should know about?
6. How should you incorporate Web Services into your technology strategy and your business strategy?

WHY ARE WEB SERVICES USEFUL IN DESIGNING CUSTOMER-ADAPTIVE BUSINESSES?

It's important for you to know and to understand what's beneath the hype of Web Services. Why are Web Services so promising? What problems do they solve?

Service-Oriented Architectures (SOA) Adapt to Change

THE PROMISE OF WEB SERVICES. Web Services are promising because they make it easy to implement service-oriented IT architectures. This is an important evolutionary step in moving from brittle, hard-to-maintain software to adaptive, easy-to-maintain software. A service-oriented approach to IT architecture makes it possible to adapt to changing business conditions and customer needs. Instead of codifying business processes into hardwired applications, different steps in a business process may call for appropriate services as needed. These services can be easily swapped in and out in order to respond to changing needs.

PROBLEMS SOLVED BY WEB SERVICES. Today's Web Services can solve a number of pressing and expensive problems. They can help you:

1. Give your business customers direct access to the information, data, and functions they need in order to interact with your firm, without requiring these customers to log onto your Web site or extranet.

2. Give your distribution channel partners direct access to the functionality they need in order to serve your mutual customers better, without requiring them to log onto your Web site or extranet.

3. Give your suppliers direct access to the information and functionality they need in order to support just-in-time inventory replenishment, without requiring them to log onto your Web site or extranet.

4. Provide affordable, easy-to-implement, end-to-end application integration both within and across your enterprise boundaries.

5. Eliminate and avoid the overhead and confusion of having redundant and conflicting IT services embedded in different applications across your enterprise (e.g., log-ons, authentication, directories and profiles, transaction management, workflow management, and so on).

6. Enable teams of developers to work independently and effectively on systems that will interact, because they are working toward a common set of interfaces rather than having to synchronize processing.

Business processes that are comprised of Web Services will be much easier to adapt to changing customer needs and business climates than are today's home-grown or purchased applications.

Web Services Help You Adapt Your Business to Changing Circumstances

Web Services—whether built upon Microsoft's .NET platform, IBM's WebSphere, BEA's WebLogic, Sun's SunONE, or any of dozens of Web Services platforms that are pouring onto the market—offer the promise of much more adaptive computing infrastructures.

Once Web Services are a reality in our IT shops and in the software we purchase, it will become easier and faster for our IT organizations to combine applications or parts of applications into new intra- and inter-organizational business processes to meet customers' changing scenarios.

Think of Web Services as self-organizing cells that can be combined into new adaptive organisms. Using Web Services, you can convert your currently brittle, hard-wired business IT infrastructure into a more flexible, adaptive, self-organizing business IT infrastructure.

In other words, each Web Service is like a cell that performs certain functions. Each cell can request information from every other cell. And cells can be combined temporarily (or permanently) in order to provide some composite functionality. If you don't know what a cell does, you can ask it. If you need something from it, you can make a request and get an answer.

You can instruct these Web Services/cells to combine themselves into business applications (like sales force opportunity management, order entry, or inventory replenishment). And then you can recom-

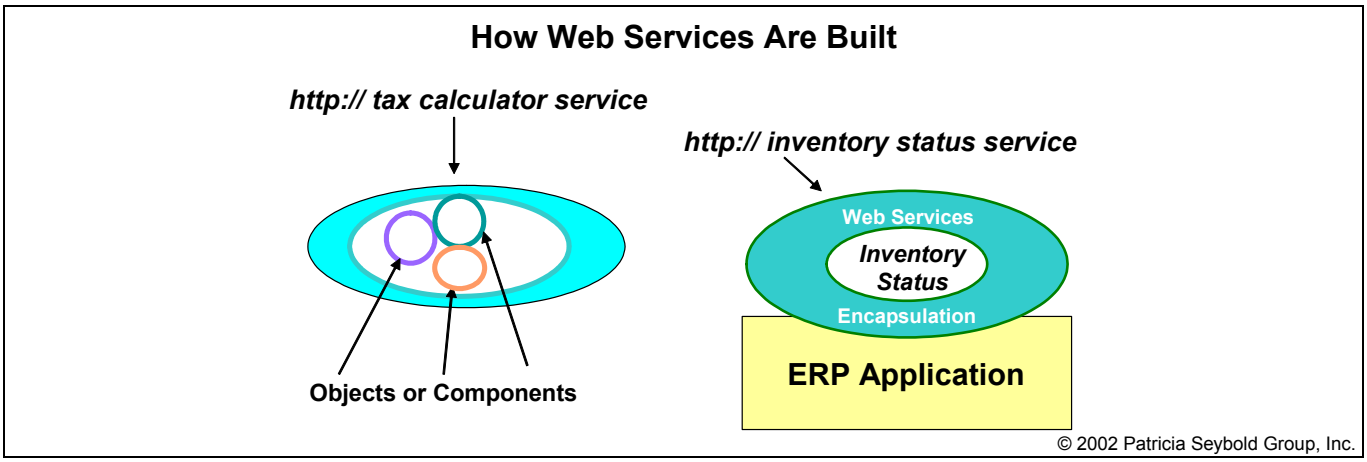

Illustration 1. Web Services are formed by putting an XML wrapper around a set of software functionality. The wrapper describes what services the bundle provides. You can create Web Services from scratch; you can encapsulate functionality in your existing applications as a Web Service; and you can buy Web Services frameworks and platforms that provide a set of Web Services infrastructure services.

bine them into larger entities that may be end-to-end business processes, such as supply chain management.

Once you've designed a Web Services-based business process, it's not cast in concrete; you can call or substitute different services as it's running. Business processes that are comprised of Web Services will be much easier to adapt to changing customer needs and business climates than are today's home-grown or purchased applications.

WHAT ARE WEB SERVICES?

Here's a high-level description of Web Services:

- A Web Service is a URL-addressable software resource that performs functions and provides answers.
- A Web Service is made by taking a set of software functionality and wrapping it up so that the services it performs are visible and accessible to other software applications. (See Illustration 1, How Web Services Are Built.)
- A Web Service can be discovered and leveraged by other Web Services, applications, clients, or agents. (See Illustration 2, How Web Services Are Registered and Discovered.) In other words, Web Services can request services from other Web Services, and they can expect to receive the results or responses from those requests.
- Web Services communicate using an easy-to-implement standard protocol known as SOAP (Simple Object Access Protocol). This is a flexible, lightweight, communications transport-agnostic XML protocol. (See Illustration 3, How Web Services Communicate.)
- Web Services may interoperate in a loosely-coupled manner; they can request services across the 'Net and wait for a response. The advantage of this approach is that Web Services don't have to be programmed with specific linkages to each other in order to work together. (See Illustration 4, How Web Services Interoperate.)
- Web Services may be combined to create new services. (See Illustration 5, How Web Services Can Be Combined to Create a New Web Service.)
- Web Services may be recombined, swapped or substituted, or replaced at runtime. In other words, a service that is comprised of a set of Web Services may use a different set of comparable services each time it runs. Or some aspect of a Web Service can be changed at runtime, such as which communications protocol it uses. (See Illustration 6, How Web Services May Be Recombined Dynamically at Runtime.)

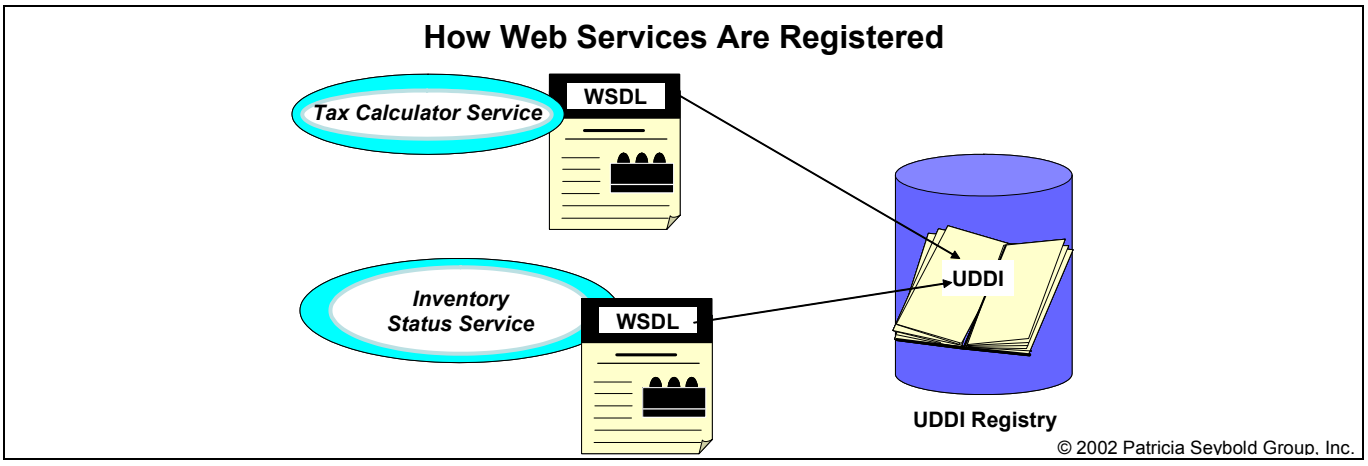

Illustration 2a. When you want to make a Web Service available, you "publish" it by providing its description in an XML document using the standard Web Services Definition Language (WSDL). To make it easy for developers (and applications) to locate Web Services, you can place these WSDL descriptions in a shared private or public UDDI (Universal Discovery, Description, and Integration) registry. Think of this as a "yellow pages" of Web Services.

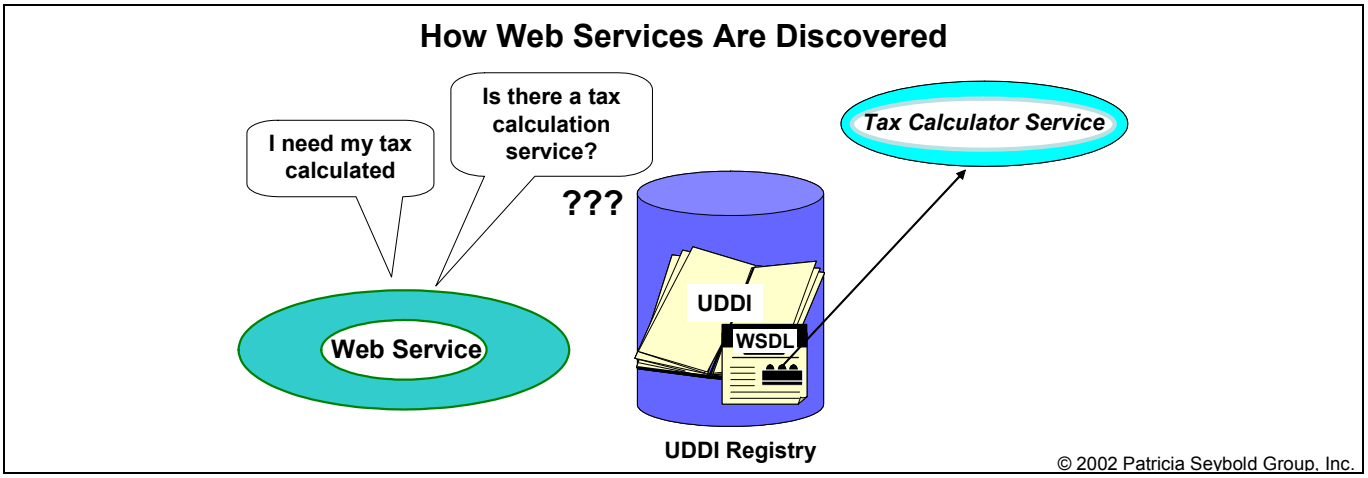

Illustration 2b. Developers who want to use a Web Service (or the Web Services themselves) may be given the URL and WSDL of a particular Web Service to use. Or they can search for an appropriate Web Service by sending a query to a UDDI Registry. They'll find any service that meets the parameters they requested. The WSDL description will include a link to the actual Web Service, which may be located anywhere on the Internet.

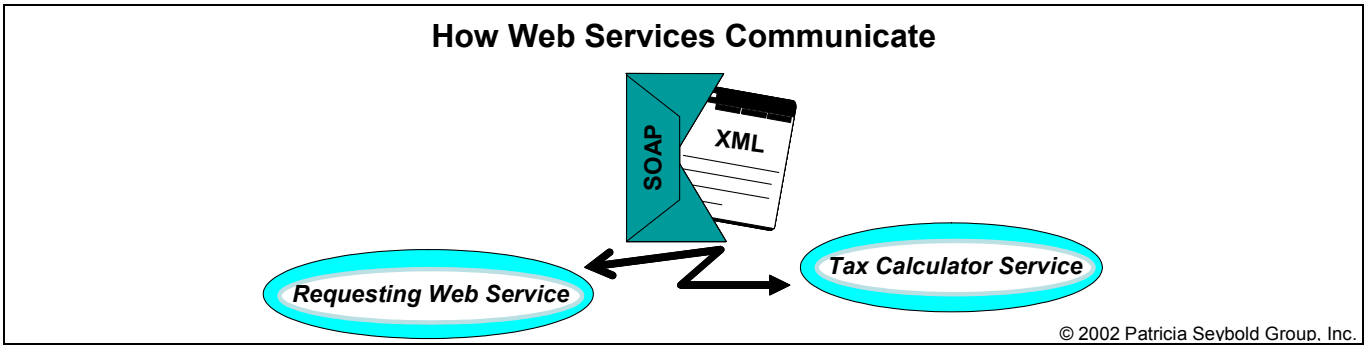

Illustration 3. A Web Service that needs functions that can be provided by another Web Service sends that request as an XML document in a SOAP (Simple Object Access Protocol) envelope. This protocol can work across a variety of transport mechanisms, either asynchronously (send the request, hang up, receive an answer later) or synchronously (send the request, stay on the line, and wait for the answer).

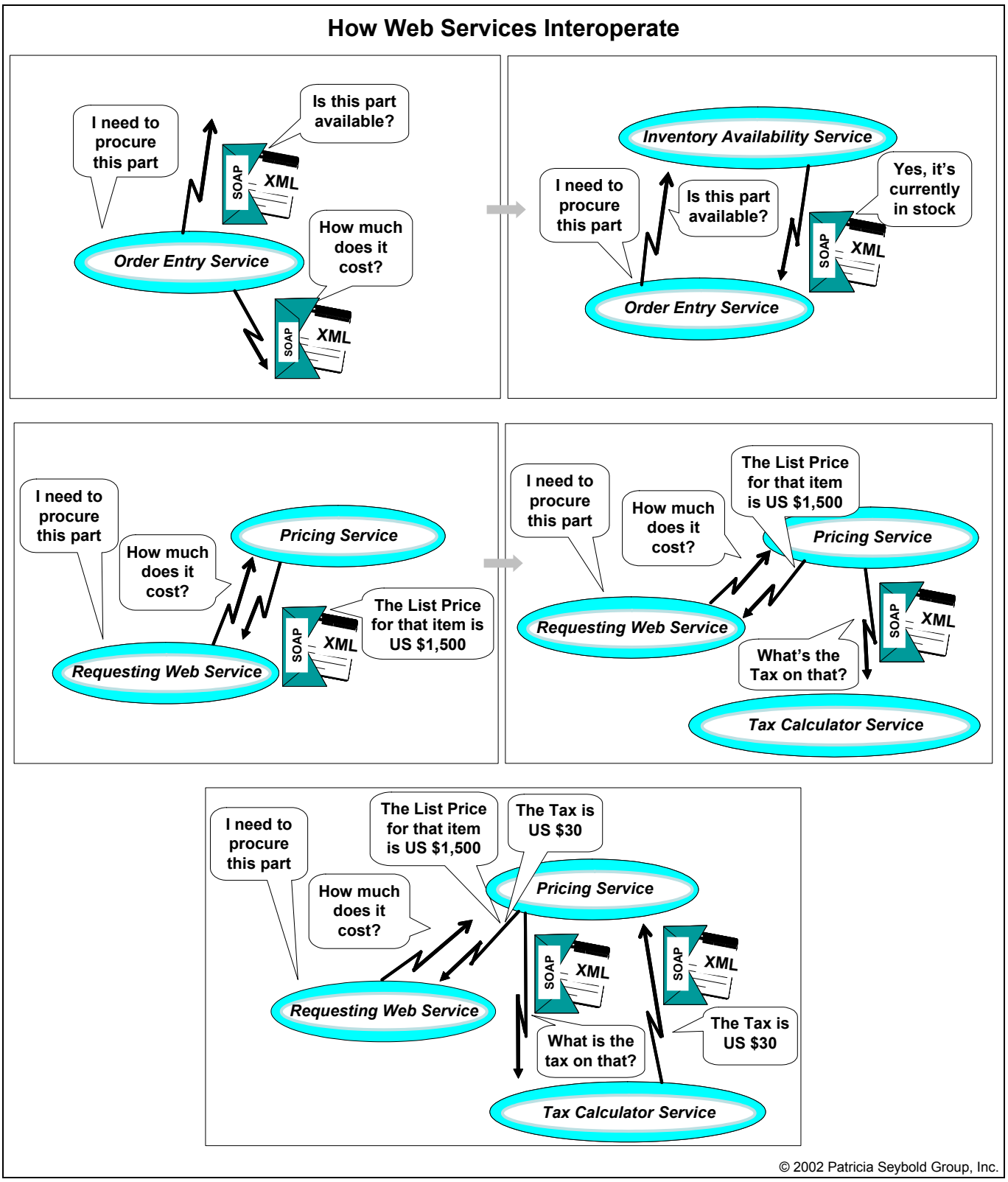

Illustration 4. Web Services may make requests of multiple services in parallel and wait for their responses.

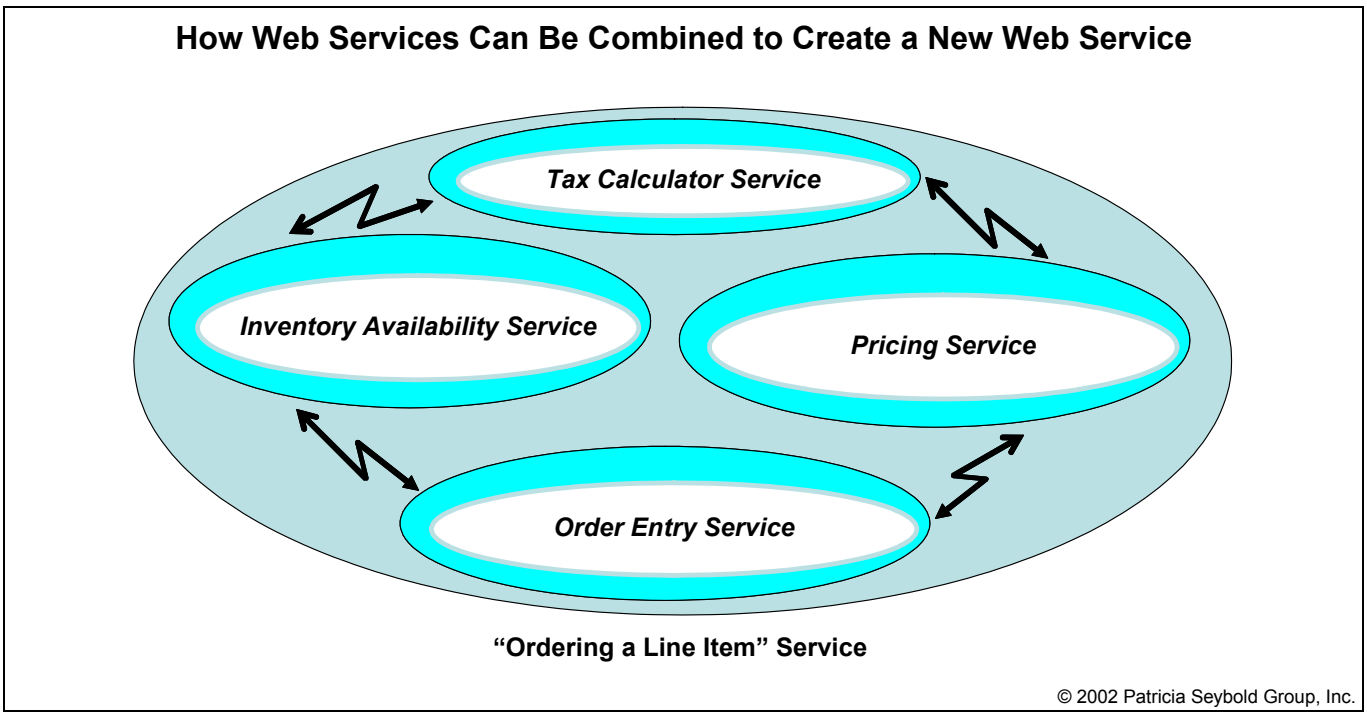

Illustration 5. You can combine any group of interoperating services into a "new" service that can be reused. Note that, if required, any of the services that make up the composite service could be replaced with another one each time it runs.

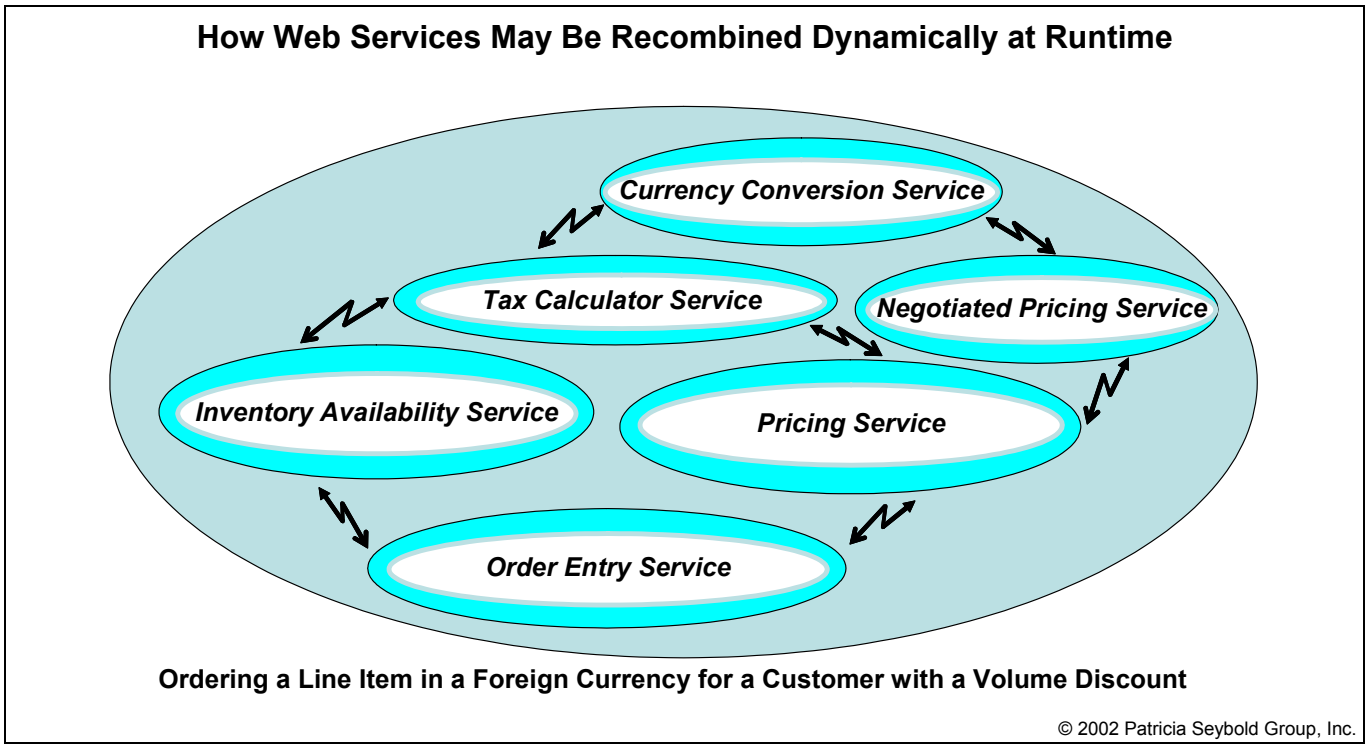

Illustration 6. You can also add new services at runtime in order to handle new circumstances or "exceptions." In this case, we need to obtain the pricing based on pre-negotiated volume discounts for this particular customer, and we need to convert the currency to the currency in the country to which the product will be shipped.

WHAT ARE THE DISTINGUISHING CHARACTERISTICS OF WEB SERVICES?

What are the things you might want to know about Web Services that differentiate them from some other forms of software building blocks.

THEY'RE SELF-DESCRIBING. A Web Service is a set of software capabilities and related information that is encapsulated and described in a programming-language-neutral format. And each Web Service explains what services it provides.

Web Services are described using XML standards. XML (eXtensible Generalized Markup Language) is independent of any programming language. XML is a standard meta-language used for describing data. Like HTML, XML is text-based and easily readable by humans. But it's actually meant to be read and understood by other Web Services. XML is used to define tags and attributes for each Web Service, using an agreed-upon set of open standards, known as WSDL (Web Services Description Language—pronounced *wizdel*).

The bottom line: no matter what programming language was used to create the functionality that is now being exposed as a Web Service, any Web Service (or XML-literate human being) can understand what functionality or capability is being offered, how to make a request of that service, and what kind of answer or results they will receive back.

THEY'RE EASY TO LOCATE & USE. When a Web Service is "published" (made available for use by other services), it is registered in a UDDI (Universal Discovery, Description, and Integration) Registry. This may be an internal directory, a directory that can be accessed by authorized business partners, and/or an external, public directory. Think of the UDDI Registry as a phone book. It lists Web Services by name and owner, offers the WSDL definition to describe the data and methods of the Web Service, and also provides a link to the Web Service URL. Early applications sometimes bypass UDDI, simply addressing a Web Service by its URL.

THEY CAN COMMUNICATE AND INTEROPERATE FLEXIBLY WITH ONE ANOTHER. What makes Web Services more flexible and adaptable than other, earlier forms of distributed computing is the fact that they are loosely-coupled services. Instead of pre-compiling the connectors or "stubs" between services as we have needed to do in the past with Remote Procedure Calls (RPC), Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA), and Distributed Component Object Model (DCOM), Web Services use a loosely-coupled interaction model. The details of the specific communication between Web Services can be either predetermined and/or renegotiated at runtime if necessary.

The information required to use a Web Service is embedded in its WSDL. The requestor's connector to the Web Service is generated from the WSDL at run time; the implementation logic is transparent (on both sides of the conversation); the location of the Web Service is immaterial. Both the Web Service and the requestor can tolerate changes to the stream of data. All of these are benefits of the loosely-coupled nature of Web Services.

THEY CAN BE COMBINED AT RUNTIME. One Web Service can request services from another Web Service. So it's possible to assemble an application or a business process by linking together a series of Web Services. It's even theoretically possible not to know which exact Web Services will be used when the application or business process is run, since each Web Service could request services from several "competing" Web Services as long as each one offers the services requested.

Today, most Web Services implementations don't take full advantage of this dynamic binding capability. Developers and architects don't yet fully trust the "other guy's" Web Service to behave in a predictable, reliable manner (even if the other guy is within their own organization). But there will be "trusted" organizations' Web Services that you may want to be able to select based on the specific context at the last minute. For example, both Federal Express and UPS offer Web Services today that enable you to launch a shipping or logistics process. You could certainly design a Web Services-based application that would select the most appropriate shipping service based on each set of circumstances, or one that would query shipment status from the appropriate provider.

Although it will take a while before the world is filled with trusted Web Services from which to choose, this loosely-coupled and dynamic-binding nature of Web Services is exactly what we need to

build dynamic, adaptive business processes that run across the Internet and onto mobile, wireless devices.

HOW ARE WEB SERVICES USEFUL IN BRIDGING THE GAP BETWEEN BUSINESS PEOPLE AND TECHNOLOGISTS?

There's one additional characteristic of Web Services that probably underlies the fact that this IT buzzword has quickly become appealing to business people. Business people have experience accessing Web Services.

Web Services are Easy for Business Folks to Understand

Business people are at least implicitly aware that the Web enables them to roll up their sleeves and to interact directly with the software applications that are used to run their suppliers' businesses. The difference between using functionality on a Web site and using a Web Service is the fact that a person doesn't have to be the one interacting with the Web-accessible resource. Instead, any software program can invoke and use (or consume) the service by requesting it over the Internet.

The difference between using functionality on a Web site and using a Web Service is the fact that a person doesn't have to be the one interacting with the Web-accessible resource. Instead, any software program can invoke and use (or consume) the service by requesting it over the Internet.

PEOPLE ARE ACCUSTOMED TO ACCESSING SOFTWARE FUNCTIONALITY VIA THE WEB. Service-oriented architectures would have remained an esoteric concept, of interest only to technology architects, if the Web hadn't become a user-friendly place to access computing resources. It's the "Web" in Web Services that makes this concept understandable.

Anyone who has searched using Google, gotten directions using MapQuest, ordered a book on Amazon, checked the status of their shipment on FedEx.com or UPS.com, or interacted with Dell, Cisco, Charles Schwab, Wells Fargo, or any other company online, has had first-hand experience with Web Services.

Anyone can access information, perform transactions, and interact with business applications via the Web. So the notion of a Web Service—some functionality or capability that can be accessed via the Web—is easy to understand.

Business People Describe Events & Activities; Technology Architects Identify the Services Required

Why are Web Services useful as a business concept and construct? Most business people can easily describe their business as a set of activities. For example, we take orders, we ship products, we send bills. We fly airplanes, we serve meals, we transport people and packages from one point to another. It's not hard for business people to appreciate that, in order to support these activities, we rely on a set of underlying services.

Some of these services are provided by internal resources; other services are provided by trusted external resources. For example, in order to accept orders, we need an order-entry service. In order to ship orders, we may need a third-party logistics and delivery service as well as a package-tracking service.

When business people describe a set of activities, technologists can easily visualize the services required to support those activities. It therefore becomes quite easy for a group of business people and technologists to draw pictures of their business consisting of activities that need to be performed and the services that are needed to support those activities.

HOW TO USE WEB SERVICES TO ADDRESS BUSINESS REQUIREMENTS

Here's a concrete example. A business person might say, "My customers want to be able to easily custom-configure their products and to place orders based on their pre-negotiated discount structures, preferred configurations, and other terms and conditions for their respective firms. Some of our best

customers want to be able to do this from within their own internal procurement systems. Other customers are willing to come to our extranet to configure products and to place orders. Still others want to place their orders through their distribution partner (such as a Value-Added Reseller or VAR) of choice. But they all want and need this custom configuration and ordering capability. To the business person, these are three similar customer scenarios—scenarios that vary based on the interaction touchpoint(s) and distribution channels that are appropriate for each different customer's context.

The technologist hearing or seeing these customer scenarios and/or descriptions of business activities might immediately visualize (see Illustration 7) the need for:

- An **authentication** service (a way to identify and authenticate the person placing the order)
- An **authorization** service (a way to determine what prices, discounts, and configurations that person is authorized to use)
- An **account profiling** service (to look up the pricing rules, volume discounts, and terms and conditions for each account)
- An **inventory availability** service (to determine whether the products or components requested are available in the desired quantities and locations)
- A **product configurator** service (to help the customer design a custom-configured product based on a set of rules and constraints)
- A **dynamic pricing** service (to calculate the correct price based on the account, the volume, the geography)
- A **tax calculation** service
- A **customs duty calculation** service
- A **currency conversion** service
- An **order entry** service
- An **order processing and management** service

Moreover, using this services-oriented architecture approach, the technologist would quickly notice three things:

1. We already have most of these services—they are provided by a variety of in-house applications. We may only need to develop and/or purchase one new service (e.g., the product configurator).

2. We can link these existing services together in order to make it easy to accept customized orders from customers in companies with different negotiated pricing, configuration options, and terms and conditions.

3. Once we've designed this new "take a customized order from a customer" service, we can make that identical service available in a number of ways. It can be accessed from our public Web site. It can be used on customer extranets for our large accounts or on our partner extranets for the partners' serving our mid-tier accounts. And, for those large accounts that prefer us to link directly to their internal procurement applications, we can tunnel through each customer's firewall and plug the service into his pre-existing procurement application. The latter will be easy to do if the customer's procurement application is able to interact with a Web Service.

Take Advantage of Web Services' Re-Usability

One of the greatest business and technical benefits of Web Services is the ability to leverage each one in a number of ways. Here are some simple examples:

SUPPORT BOTH APPLICATIONS AND PEOPLE WITH THE SAME WEB SERVICE. As we saw in the example above, you can use the same Web Service (e.g., an inventory availability service) to perform the same functions in a number of places in parallel. To wit:

- On a public Web site
- On a private Web site (extranet)

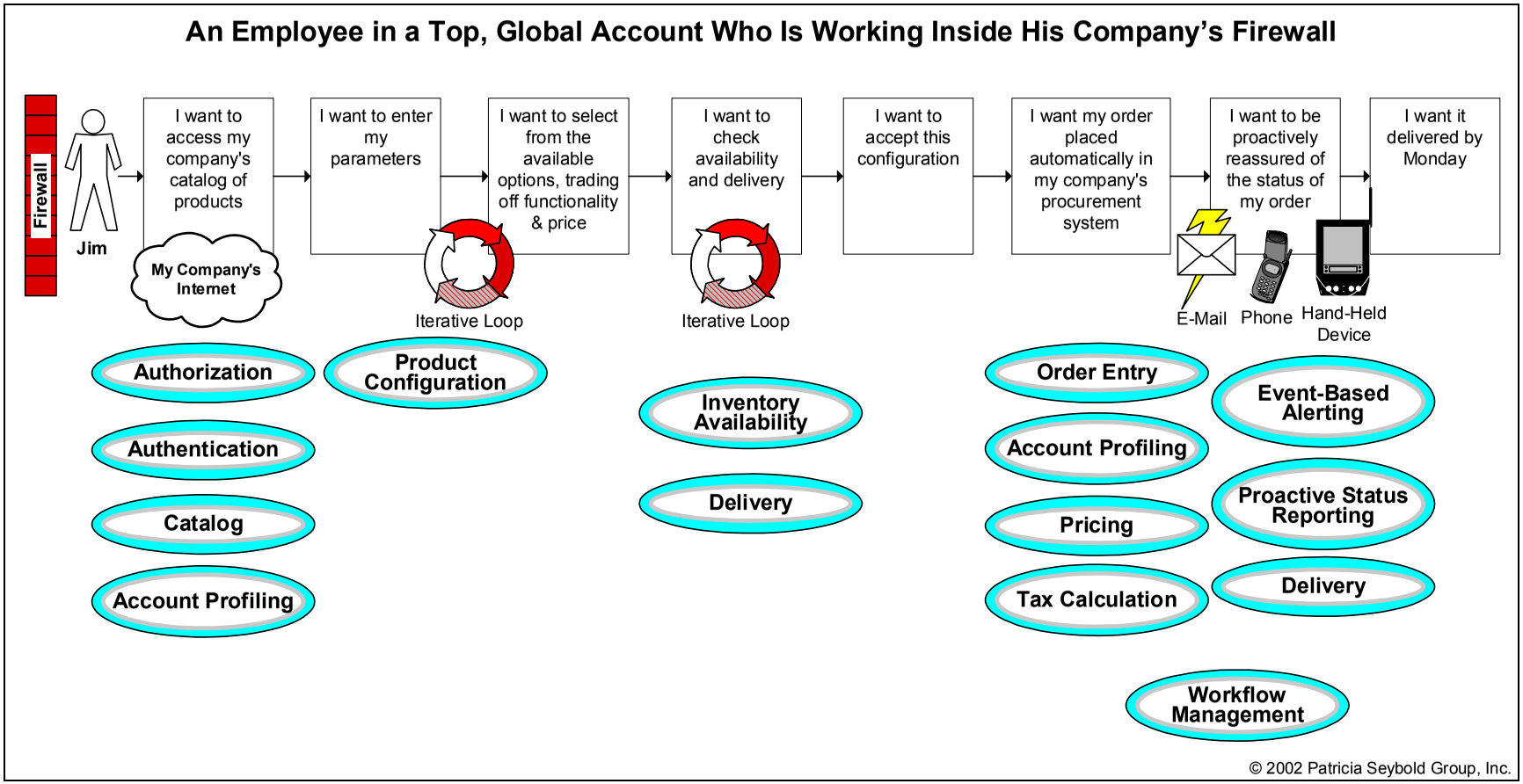

Illustration 7a. An employee in a top, global account who is working inside his company's firewall. Jim wants to place an order for a new computer that he needs to have delivered by Monday. He expects to be able to access accurate, real-time information about the product options, pricing, and availability without ever leaving the comfort, safety, and convenience of his company's intranet. He wants to access his company's internal multi-supplier catalog and to be able to select from among the company-approved models and options. Then he wants to custom-configure his preferred model and check shipping and availability to be sure that this configuration can be delivered by Monday. He expects to receive his company's pre-negotiated pricing, based on the volume discount schedule they have in place with the approved supplier. He also expects to be able to "flow" the order from his intranet catalog, through the vendor-supplied configurator, into his company's own procurement application and approval workflows without rekeying anything. And, of course, he wants to be proactively notified about the status of the manufacture and delivery of his PC.

An IT Influencer in a Mid-Sized Account Who's Ordering from the Supplier's Web Site

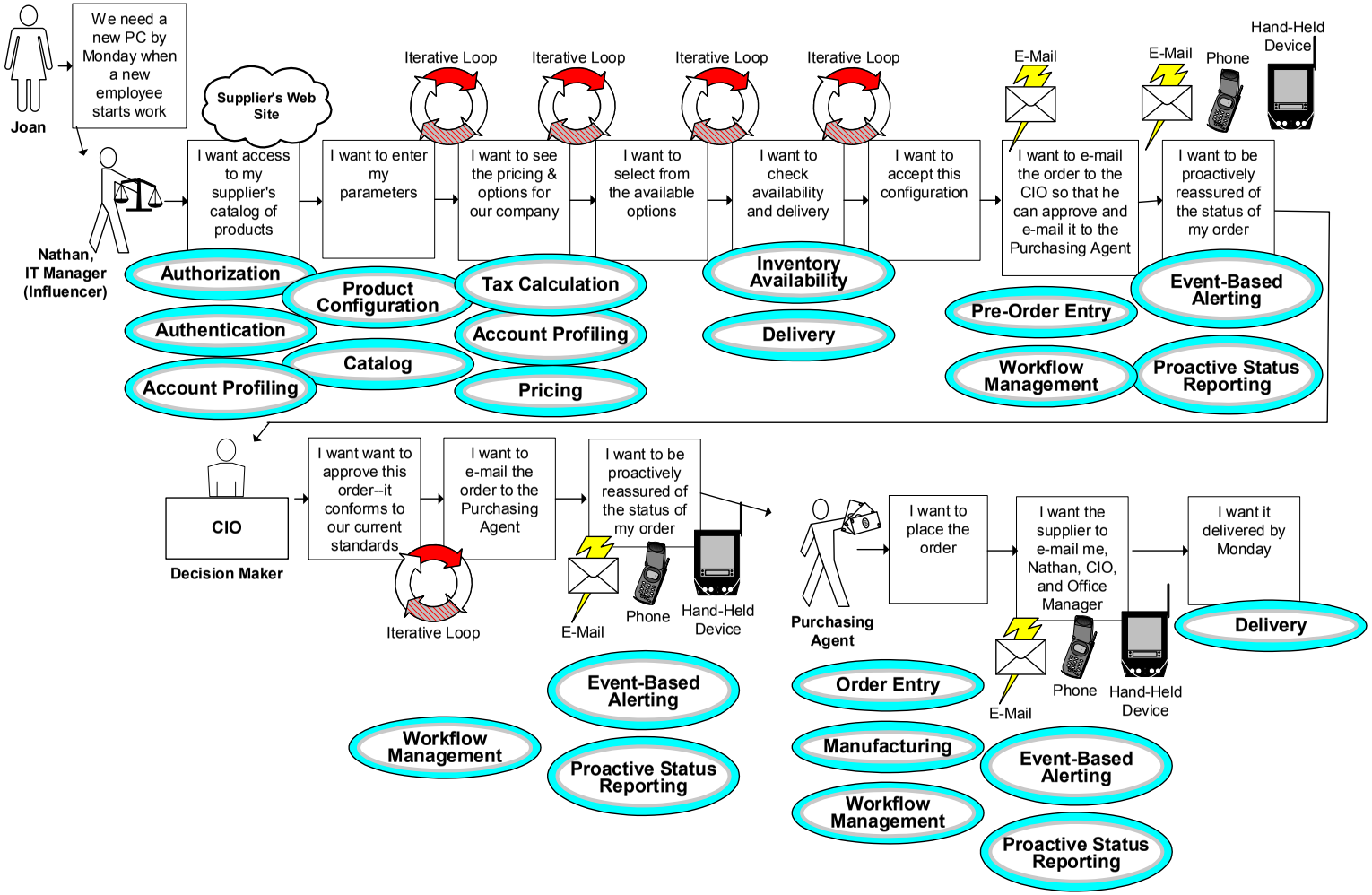

Illustration 7b. An IT influencer in a mid-sized account who's ordering from the supplier's Web site. Nathan is the guy who is on top of the latest technologies that will still meet his company's standards (since he helps set those standards), so he uses his current preferred supplier's Web site to research, configure, and pre-order. He expects to receive his company's negotiated pricing and volume discounts. He also expects to be able to route the pre-configured order to his boss, the CIO, for approval and onto his company's purchasing agent to complete the order. Everyone who was involved in the approval/purchasing workflow may choose to be proactively notified about the order and delivery status, since they know that Joan, in the HR department, is counting on having the new PC available when a new employee starts work on Monday.

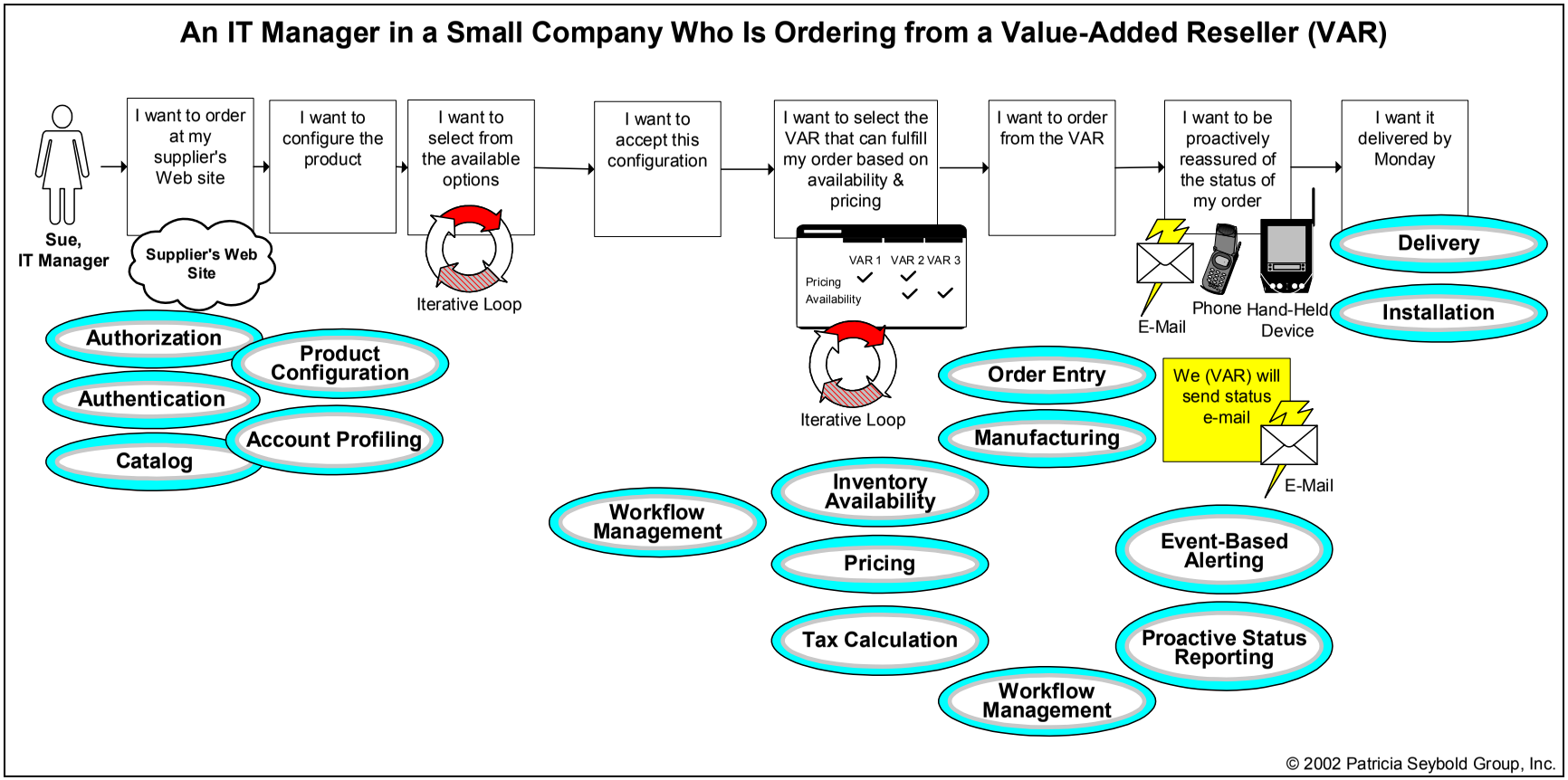

Illustration 7c. An IT manager in a small company who is ordering from a value-added reseller (VAR). Sue is the IT manager for a small company. She knows what she needs—a PC by Monday—and she goes to her preferred supplier's Web site to order and configure it. Since she relies on her local VAR for installation, configuration, and integration into her network, she wants to configure the product she needs, but then place the order through the VAR who will deliver and install it on Monday. She expects a seamless integration between her supplier's Web site and those of her preferred VARs. She does not want to rekey anything. She will make her decision about which VAR to use based on which one can actually provide the desired machine by Monday at a reasonable price. She expects the VAR to proactively notify her about the order status and the delivery and installation scheduling.

- From a customer/partner/or supplier's application behind their firewall
- As a portlet in an employee, customer, or supplier portal

CREATE A WEB SERVICE FROM AN EXISTING APPLICATION AND MAKE IT AVAILABLE TO OTHER APPLICATIONS. Every company has a set of production applications. These are the ones that run your business. This is the first place to look for candidate Web Services that can be re-used and re-combined. Don't think of your entire legacy application as a single Web Service. Instead, think of the various functions it performs, e.g., order tracking, inventory management, credit-card processing, currency conversion, account reconciliation, production scheduling, and so on. Each of these capabilities is a candidate Web Service. What's nice about them is that they are already production-tested, operational functions. By turning them into Web Services, you can make them more readily available to be re-used by other internal and external applications. So you can unbundle a service from an existing application and re-use it to support a number of applications.

CREATE AND/OR PURCHASE A SET OF RE-USABLE WEB SERVICES. There are probably a number of infrastructure services that you'll want to build or buy and then use over and over again to gain the most consistency and maintainability across applications. Examples of these might be services such as single sign-on (authentication and authorization), transaction management (commit, roll-back), and user profiling (directory services, identity management).

These Web Services infrastructure services constitute the main battleground for the major suppliers of Web Services deployment platforms, such as BEA's WebLogic, IBM's WebSphere, Sun's SunONE, and Microsoft's .Net framework.

At the same time, Oracle, PeopleSoft, Siebel, and SAP are also embracing Web Services. So if you have already invested in one or more applications from these suppliers, you may want to re-use some of their evolving Web Services infrastructure to support your new applications. Don't think of these as application suites. Think of them as portfolios of Web Services.

For example, all of these players offer some form of workflow management service. It's probably a good idea to select one or two of these to use over and over again instead of finding yourself with five different workflow management services, each with a different, incompatible business rules engine.

CREATE NEW FUNCTIONALITY AS RE-USABLE WEB SERVICES. Since Web Services lend themselves to re-use and leveragability, it stands to reason that if you're developing new applications, you should probably create them as modular Web Services. But, in doing so, you'll want to be sure that the designers and developers involved have experience in the design of loosely-coupled applications. If they don't, the chances are they'll use Web Services tools to develop the same tightly-coupled software applications they've written in the past.

WHAT ARE THE PITFALLS OF WEB SERVICES?

What do you need to know about the downside of Web Services? From a technical standpoint, there are some performance trade-offs to worry about. There are choices to be made about when to use asynchronous versus synchronous processing, for example. Another technical issue revolves around auditability. You need to be sure that the Web Services themselves, and the environment in which they're deployed, contain the instrumentation that will allow them to be monitored in real time. How else will be you able to trace and track what's happening in this much more dynamic networked world?

Creating a Web Service is a proverbial piece of cake. Today's software development tools make it easy for programmers to wrap up chunks of code and to publish them as Web Services. But those tools don't insure that the services that have been created are well-architected for the loosely-coupled, dynamic-binding world of a services-oriented architecture. The primary difficulty in creating Web Services and applications with a service-oriented architecture is getting developers and architects to think *loosely coupled* and *asynchronous*. The transactions that client/server developers imagine are tightly coupled and synchronous: it's a two-way conversation, and we're both staying here until we're both finished.

For that reason, you're going to want to have a small team of senior architects lead your Web Services development and deployment initiatives. Designers who have been developing and deploying distributed object systems for a number of years are your best bet. You'll need to find the few souls within your IT organization who are the most experienced in designing loosely-coupled asynchronous applications and have them do the design work, educate the rest of your development team, and do quality assurance on the Web Services you develop in-house or purchase or use from third parties.

WHAT SHOULD YOUR WEB SERVICES TECHNOLOGY STRATEGY BE?

Select your team of IT architects (from one to six people, max). Start with applications architecture. Take the applications and software architecture you have running today and draw a picture of how you'd decompose it into logical services. Decide which of those services are currently robust and can be leveraged by other applications.

We believe that the most strategic use of Web Services is to select robust, operational functionality that is already tested and deployed and to expose that functionality as a Web Service to customers, partners, and/or suppliers. In others words, don't start from the inside out. Start from the outside in.

Then map out the new application and business process activities and events that are in your pipeline and on your radar. Determine what existing services will be required to support them. Decide what new services you'll need.

Gather end-customer and stakeholder requirements in the form of Customer Scenarios. What are the outcomes that customers need? What are the tasks they're happy to do in order to meet those needs? What resources and business processes will support each of those tasks and activities? And what services will be required to support those business processes? Since customer scenarios are likely to change on the fly (as customers' contexts and perceived needs change), starting with customer scenarios will force you to design your services to be loosely coupled and interchangeable.

Once you've done this high-level design work, you'll be ready to evaluate Web Services deployment platforms and development tools. And you'll be better able to vet alternative packaged applications based on your services-oriented architecture.

Now focus on the core infrastructure services that you'll need to underpin many applications, business processes, and customer scenarios. Here are some candidates for you to consider. We think that the chances are pretty good that the first set of shared Web Services you'll want and need will be (see Illustration 8):

- **Directory Services**—to manage the information about users, customers, and resources as well as a registry to locate and store Web Services.
- **Authentication Services**—to manage the identification and certification of identity for all internal and external people, applications, and other Web Services.
- **Authorization and Entitlement Services**—to manage the rights and permissions of each individual or requesting application or Web Service. What are they allowed to know about, to see, to request services of? Which services are they entitled to interact with and in what ways?
- **User Session Management Services**—what's the current state of the set of services to which the end user or application has made requests? Since Web Services may be "sitting around" waiting for responses from other Web Services, there needs to be a way to gain visibility into the state of those requests.
- **Logging and Reporting Services**—to create an audit trail of what happened and to enable you to identify and fix exceptions. It will help to have a common format for this information that everyone is familiar with.

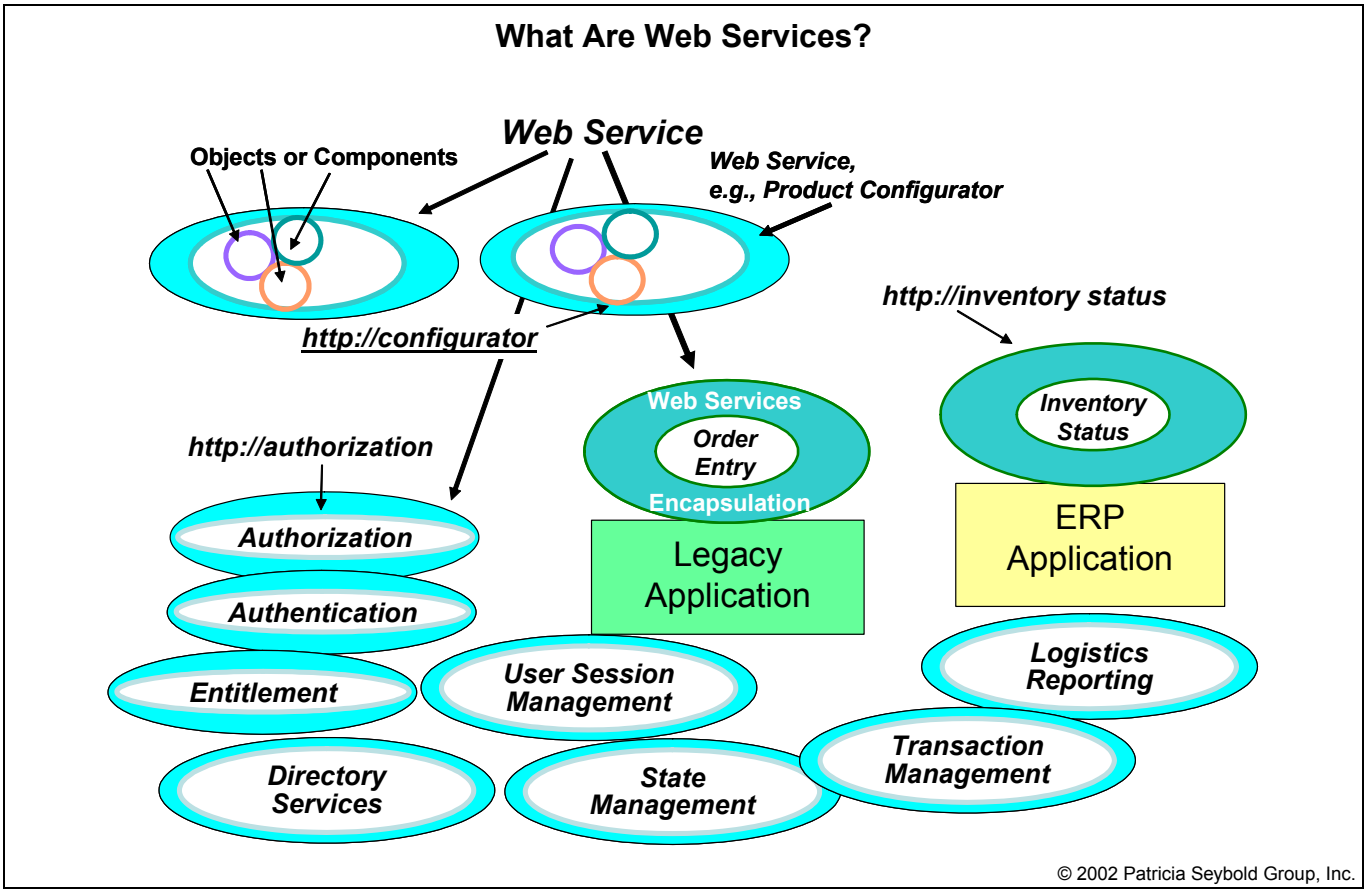

Illustration 8. You can develop Web Services from scratch by combining software components. You can encapsulate existing functionality from pre-installed applications as Web Services. And you can purchase frameworks that provide commonly used Web Services, such as Authorization, Authentication, Directory, Entitlement, User Session Management, and Logging and Reporting Services.

Then, of course, it's time to select one or more projects and begin to design, develop, implement, and deploy Web Services. The common wisdom is that you should begin this activity internally, by using Web Services to integrate internal applications—another way to approach EAI. We disagree. We believe that the most strategic use of Web Services is to select robust, operational functionality that is already tested and deployed and to expose that functionality as a Web Service to customers, partners, and/or suppliers. In others words, don't start from the inside out. Start from the outside in.

WHAT SHOULD YOUR WEB SERVICES BUSINESS STRATEGY BE?

We believe that you should use a Web-Services approach to give you rapid time-to-market in connecting your customers, business partners, and suppliers to your core operational functions. Don't try to do end-to-end application integration using Web Services until you know exactly where the flexibility needs to be designed into each end-to-end business process.

Start by Exposing Your Core Operational Functionality to Key Stakeholders

Instead, identify the core capabilities that your customers and partners need today. There are a host of them: order status, delivery status, order placement, inventory availability, pricing, service activation, billing inquiry, and so on. The good news is that you've probably already exposed many of these services on your Web site(s) or portals. Now you need to re-architect those Web access mechanisms to

take advantage of Web Services. As you do so, you'll gain a triple whammy. You'll have a more robust portal/Web architecture. You'll be able to offer this functionality as Web Services to customers, partners and suppliers. And this will save you time and money doing one-off integrations with your external stakeholders.

Don't Do Business Process Integration; Instead Support Customers' Scenarios with Critical Web Services to Yield Adaptive Business Processes

Once you've exposed some of the functionality of your operationally-tested and robust applications as Web Services, you're ready to proceed to the next step: designing adaptive business processes. We define Adaptive Business Process Management as "*the ability of organizational processes to respond to the specific contexts of the customers' changing scenarios*." Notice that we believe that businesses and business processes need to be highly adaptive in order to be effective in the customer economy.

We believe that you should use a Web-Services approach to give you rapid time-to-market in connecting your customers, business partners, and suppliers to your core operational functions.

So, be careful! There's a lot of hype swirling around the "new" field of business process design and management in conjunction with hype surrounding Web Services. The Web Services and EAI marketing pitches give you the impression that if you could just "get it right"—if you could just design your end-to-end business processes correctly once and for all—you'd be home free. And, if you could use that business process design as a blueprint and LEGO™-kit to integrate your applications, you'd have rewired your business to be more efficient and more effective.

We believe that logic is flawed. Business processes are internal constructs designed to produce consistent, predictable results with little variation. Customer scenarios are external activities that are designed to yield a desirable outcome with the best possible customer experience. Customers don't often get through an entire scenario (e.g., refinance my home, plan a business trip, install a new assembly line) without several changes in context and iterative decision-making. It's neither possible nor desirable to "hardwire" an end-to-end customer scenario. But it is desirable to be able to support each iterative step in a customer scenario with consistent, repeatable, yet flexible and adaptive business processes.

If you begin your business process design and applications integration design by capturing as many key customer scenarios as possible, you'll be able to identify the core services that are required over and over again to support these scenarios successfully. You can then prioritize to your IT department which Web Services you need the most urgently in which order, so that they can to begin to deliver a dynamic Web Services-based infrastructure that will adapt flexibly to changing customers' scenarios.

Why are business processes that are constituted from Web Services more adaptive than business processes that rely on a single application or on two or more tightly-integrated applications? When you design a business process by stringing together a set of Web Services, each of which is requesting services from the others, it's easy to introduce or to substitute a new service at any time. One Web Service may be substituted for another Web Service at runtime as long as it fills the request specified according to the conditions of satisfaction stipulated by the requesting service.

Identify Valuable Services You Can Offer

Finally, from a business strategy standpoint, it's not too soon to begin conceiving of your business as a set of services that that can be combined and recombined to deliver value to a wide range of stakeholders. Remember that anything your business is currently doing using software may be useful to offer as a Web Service. Oft-cited examples of software services that are already available as Web Services include:

- Credit-checking
- Tax calculation
- Shipment tracking

These are simple examples. But what are the services your firm currently provides that could be made available to others as valuable Web Services?

Think about the legacy applications in which you've invested over the years. Which of those applications incorporates business know-how and processes that have given you a competitive edge?

Think about the brand new capabilities you're developing today in order to give you a competitive lead. The chances are those are precisely the capabilities you'll want to think about packaging up as Web Services in order to gain greater leverage from those investments.

You should be thinking about these today in order to be ready to be able to roll them out within the next three years. In fact, the most forward-thinking companies are already taking their "crown jewels," exposing them as Web Services, and selling them as services to their former competitors.

An Executive's Guide to Web Services

Anatomy of Web Services

Five Technology Categories and Selection Criteria

By Susan E. Aldrich

NETTING IT OUT

Business executives responsible for customers, channels, supply chain, and product strategy see exciting opportunities in Web Services. Early projects are demonstrating the value of Web Services for connecting operational applications with customers and trading partners, solving a persistent and expensive set of integration problems.

IT executives, architects, and developers will select a range of Web Services technologies to support emerging business requirements and to enable forward progress on existing integration problems.

We've identified five major categories of Web Services technologies:

- Web Services Development and Assembly Environments
- Web Services Deployment Environments
- UDDI and Directory Services
- Web Services Service Infrastructure Services
- Adaptive Business Process Environments

Given the immaturity of the Web Services arena, today's products are not yet hardwired to these categories. But these categories definitely represent the succession of investments you'll make in Web Services.

Web Services can be investigated and piloted with a very small initial outlay, but this will ultimately reach into the majority of your applications and potentially claim a major portion of your integration budget.

YOU SHOULD WELCOME WEB SERVICES

Web Services Are Real and Valuable

Web Services are the new *new* thing, accordingly overhyped and accompanied by outrageous value propositions. But don't be misled; Web Services offer a tremendously effective architecture and technology set. Web Services have a critical contribution to make toward solving that intractable business and technology problem: turning stove-piped and rigid systems into adaptive business processes.

For this reason, we see a rapid adoption of Web Services capabilities. By 2006, Web Services will provide access to critical functions scattered across most of the world's business applications.[1]

The value of Web Services has already been demonstrated. There are a number of impressive Web Service-based applications in use today, some of which cross enterprise boundaries, and at least one that supports millions of users. One example is Deutsche Telekom, which uses Web Services to connect 100+ content providers with 5 million mobile phone service subscribers.

Today, we can expect the following benefits from Web Services-based applications:

- A standard way to expose legacy application functionality as a set of reusable services that

[1] See "The Web Services Freight Train," May 2, 2002, http://www.psgroup.com/doc/products/2002/5/TA5-2-02CC/TA5-2-02CC.asp.

can respond to requests for functionality or information from other applications and services

- Application-to-application integration with less pain than previous methods
- Rapid application assembly out of tested, trusted, and "guaranteed to interoperate" application functionality
- A standard way to develop new (sets of) application functionality as self-contained, self-describing services that can automatically interoperate with other services in a well-behaved, manageable way
- An accepted, standard way to develop and/or assemble Internet-native applications for both internal and extended-enterprise use

Utopia and Reality

The visionary promise of Web Services is a world where application components are assembled into services that can be loosely coupled to create dynamic business processes that span enterprises, computing platforms, and data models.

This utopia is several years away because the world has yet to finish defining even a conflicting set of standards for transaction integrity, reliable messaging, workflow, and secure interactions, among others. Lacking these standards, development teams can, of course, design reliable and secure transactions by creating and deploying proprietary extensions.

But this means that, in order to use a high-integrity Web Service, your developers may need detailed knowledge of those extensions. The organizations that consume and provide these services need to be in a relationship strong enough to warrant investing in understanding each others' processing systems.

Therefore, the reality is that the first wave of Web Services implementations will occur between trusted trading partners.

Nevertheless, organizations today are making those investments, both for Web Services deployed inside the company and for those offered outside the company.

In order to use a high-integrity Web Service, your developers may need detailed knowledge of those extensions.

Web Services Definition

Web Services present a common communication protocol and language for applications to interact. In human terms, Web Services provide a way for two applications, parties, organizations, or institutions to interact. People use the term pretty much interchangeably to describe the architecture, the technology, and applications that are built using the technology. Some of the more confused will represent Web Services as synonymous with products that enable Web Services, such as. NET. So if your head is spinning, you're in good company.

Here is a technical definition of Web Services:

- A Web Service is a URL-addressable software resource that performs functions and provides answers.
- Web Services communicate using an easy-to-implement standard protocol known as SOAP (Simple Object Access Protocol). This is a flexible, lightweight, communications transport-agnostic XML protocol.
- A Web Service is located by its listing in a Universal Discovery, Description and Integration (UDDI) directory.

OVERVIEW OF WEB SERVICES TECHNOLOGY CATEGORIES

Generally speaking, Web Services technologies you'll be considering fall into five categories. Our categories don't exactly correspond to product offerings yet, due to the immaturity of the Web Services market. Vendors and buyers still aren't sure what Web Services capabilities they need or where those capabilities should reside. While that set of issues settles out over the next 12 months, use our list as your guide.

Reading Illustration 1 from the bottom up, our categories are:

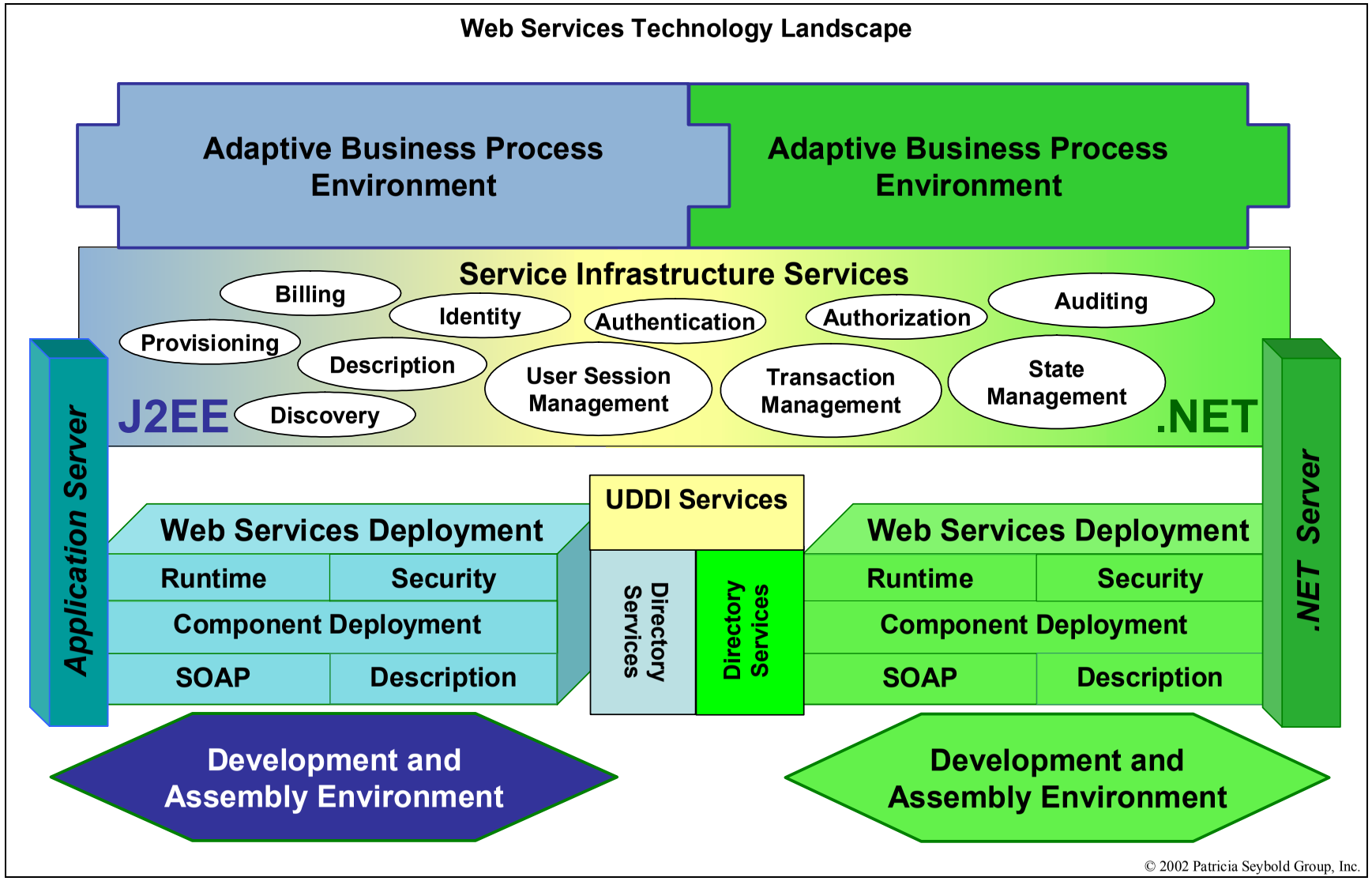

Illustration 1. The landscape for Web Services includes a range of technologies, spanning both the service requestor's environment and the service provider's environment. In this diagram, the requestor application runs in a .NET environment, and the provider's application runs in a Java environment. Our landscape entries don't exactly correspond to product categories yet, but indicate more of a continuum.

- Development and Assembly Environments
- Deployment Environments
- UDDI and Directory Services
- Service Infrastructure Services
- Adaptive Business Process Environments

Note that, in our diagram, application servers, directory services, and IDEs maintain a critical supporting role for Web Services, just as they do for all applications.

Reading the Landscape Diagram

LEFT AND RIGHT: WEB SERVICES SUPPORT A DIALOGUE. In our diagram, the left side represents a full stack of technology for Java environments, and the right side represents a full stack for .NET.

Why? There are two sides to every Web Services interaction. There is the requestor of the service, also known as the client or the consumer. There is the provider of the service, also known as the publisher.

Business relationships may mandate the adoption of Web Services.

Typically, an application that provides Web Services also consumes them in a given interaction. Think of it like a phone call; generally, the party on each end has something to say and something to ask.

Web Services dialogues can involve only Java, or only .NET. A single application might call a J2EE-based Web Service and then call a .NET-based Web Service. We depict a single J2EE stack and a single .NET stack for simplicity's sake.

BOTTOM TO TOP: WEB SERVICES STACK. In our diagram, the life of a Web Service begins with *development and assembly*, the bottom row of our diagram. Web Services are deployed into *deployment environments*, the next layer, and connect via *UDDI and directory services*. Note that there are two sets of directory services, representing the participants on each side of the dialogue. The Web Services may run in an *application server*.

Service Infrastructure Services, the next layer of the stack, represents a common set of services required by all Web Services, such as security, management, billing, state management, transaction management, and auditing.

The final layer, *adaptive business process environments*, provides the orchestration of groups of Web Services and infrastructure services into business processes.

Who Buys, Uses, or Drives the Adoption of Web Services?

BUSINESS LEADERS. Business leaders responsible for strategy, business development, product management, customer service, channels, procurement, and other IT-leveraged areas are the true drivers of Web Services investments. It is business strategy, business development, business relationships, and business operations that create the need for adaptive business processes. Business relationships may mandate the adoption of Web Services: if a key customer, supplier, or distributor insists on using Web Services to interact, you may feel you have no choice.

Business leaders are responsible for identifying the strategic value of making certain business processes more flexible and polling trading partners about Web Services plans.

IT EXECUTIVES. IT executives refine technology strategy for inter-organization interactions, identifying what services should be shared across divisions, and which application functions need to be shifted to a Service-Oriented Architecture (SOA).

IT ARCHITECTS. IT Architects establish a Web Services architecture, with a primary goal of separating common and application infrastructure services from application logic.

IT architects also establish the preferred development languages and deployment platforms. Although one of the promises of Web Services is the universality of their interactions, Web Services are created for a specific developer model (e.g., Java, .NET), and with a target deployment environment in mind.

IT DEVELOPERS AND DEVELOPMENT MANAGERS. Development tools are carefully evaluated and chosen for their impact on developer productivity. In larger development groups, IDEs are mandated, so any development tools have to plug in to corporate's IDE.

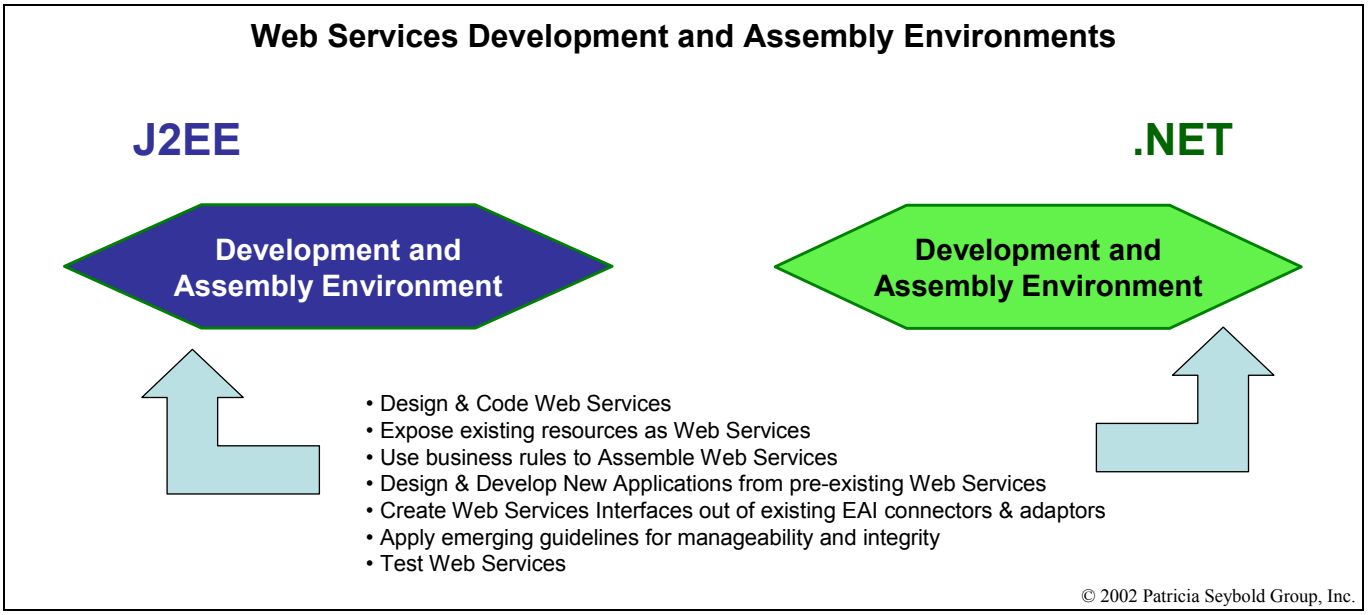

Illustration 2. Web Services development environments typically target a particular deployment environment, but the functions that need to be covered are the same for J2EE and .NET.

IT OPERATIONS. Since the programmers who choose development environments aren't typically familiar with managing the operations of the applications they create, IT operations needs to be involved in the selection process. *Deployment environments* should be carefully chosen for their contribution to Web Services application manageability, reliability, and extensibility.

WEB SERVICES DEVELOPMENT AND ASSEMBLY ENVIRONMENTS

Web Services development and assembly environments are typically chosen by IT developers, managers, or architects. Illustration 2 highlights development and assembly in our landscape.

In our view, a Web Services development environment supports these development activities:

- Design and code Web Services
- Expose existing resources as Web Services
- Use business rules to assemble Web Services
- Design and develop new applications from pre-existing Web Services
- Create Web Services interfaces out of existing EAI connectors and adaptors
- Apply emerging guidelines for manageability and integrity
- Test Web Services

We believe the choice of Web Services development environments is driven more by existing corporate investments than by the relative merits of Web Services development environments. Why?

You can expect that the popular development tools and IDEs will be extended to support Web Services. It's a safe bet that continuing to use the tools you've already invested in—training, procedures, policies, software licenses, integration with other related tools—will be far cheaper and far more comfortable than installing new development tools.

Nevertheless, we expect that there will be a major marketing and sales battle waged over the hearts and minds of developers. Every Web Services deployment platform supplier views the Web Services development environment as the critical beachhead on which to win the loyalty of developers. That's why many of the Web Services development tools are being offered free of charge by suppliers such as HP, Microsoft, Oracle, Sun, and even IBM. Micro-

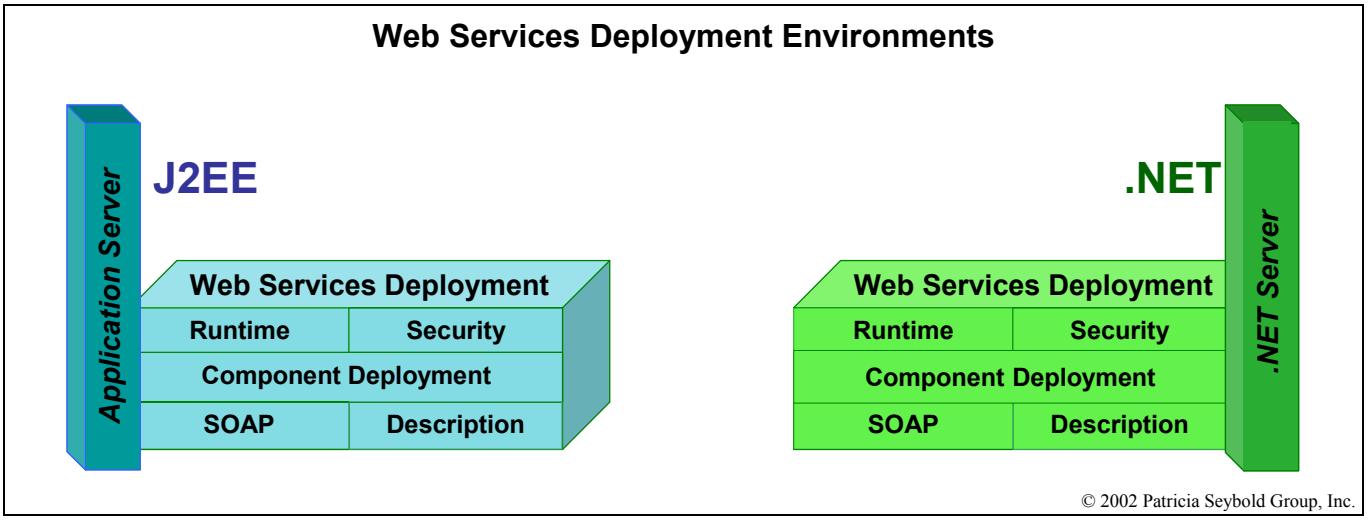

Illustration 3. Deployment products are platform specific, but the required functions are consistent. Note that the Web Services deployment functions are supported by J2EE and .NET servers.

soft is aiming at capturing all the hearts and minds by making .NET developer heaven.

There are several ways to categorize Web Services development and assembly environments:

- **Price Range**—free, mid-range, and high end
- **Scope**—tool, suite, or IDE
- **Environment—**.NET or Java
- **Platform**—Microsoft or J2EE Application Server
- **Vendor**—open source, small, mid-size, household brand name

WEB SERVICES DEPLOYMENT ENVIRONMENTS

Web Services deployment environments should be selected by IT architects and IT operations staff. In fact, development environments typically target a particular deployment environment, so the choice of deployment environment is implicit in the choice of development environment. Illustration 3 highlights development and deployment in our landscape.

Web Services deployment environments typically run in J2EE application servers, or on Microsoft .NET. Your choice of deployment environment is greatly influenced by what application servers and platforms your organization currently uses.

In our view, a Web Services deployment environment has the following responsibilities:

- **Runtime.** Provide an application runtime environment for Web Services, with configuration and lifecycle management; provide fault tolerance and recovery; log; trace; and enable performance tuning of active environment.
- **Security.** Provide runtime services for providers and requestors to verify authentication, proof of origin, message integrity, and message privacy.
- **WSDL and XML Schema Support.** Provide descriptions of all deployed Web Services for potential clients, support the XML data typing system.
- **SOAP Support.** Provide, or integrate with, a SOAP service that manages the sending, routing and receiving of SOAP messages; XML-to-language type mapping; application invocation; propagation of error messages.
- **Component Deployment.** Provide tools that will expose existing application parts (components, programs, and objects) as Web Services. This includes creating WSDL, the wrapper, and the SOAP interface for each Web Service, as

well as listing the service on a UDDI registry or other directory.

- **Management.** Monitor and report on the operation and results of Web Services' operations and performance (including logging). Enable the adjustment of parameters and constraints to improve the performance and/or quality of service of Web Services individually and collectively.

UDDI AND DIRECTORY SERVICES

Universal Description, Discovery, and Integration (UDDI) directory service should be selected by IT architects. Business leaders also have a role in specifying requirements and managing relationships in the directory services arena.

It's definitely time to shift to a single robust, scalable directory service.

The roles of UDDI and directory services technologies are locating, publishing, and securing Web Services. UDDI is a standard for publishing and locating Web Services. The UDDI directory lists the business entity, port binding, methods, and data streams for each Web Service. These directories can be private or public. Illustration 4 highlights UDDI and directory services in our landscape, creating the logical connection between providers and consumers of Web Services.

A Web Service doesn't have to be published in a UDDI directory. Many early Web Services implementations are not using UDDI as the mechanism to find or publish Web Services, but rather are listing each Web Service URL in a configuration file or a directory.

Directories come into play for identification, authentication, and authorization of the user of a Web Service. This user can be a person or another Web Service.

No doubt, your directory technology is already in place. It's fair to say that the directory is so fundamental as a technology and as a repository of corporate policy that no one wants to replace or alter it. But you'll have to face the painful fact that the directory service you will be using in the future may be different than which one(s) you're using now.

If your company is at least mid-size, you probably don't have a single directory, with a single authentication and identity management service. It's definitely time to shift to a single robust, scalable directory service. You are likely to find that Web Services will vault your directory entries from the thousands to potentially millions. Ideally, you should use the same directory technology for your employees, partners, customers, and all other users.

Of course, your trading partners have the same directory issues and pressures, and this is a great time to sit down with them and discuss directions and requirements.

Authentication and identity for inter-company interactions are thorny issues. The open questions for the business leader are:

- How do I secure interactions with another organization?
- Am I content with my trading partners' authentication and identity mechanisms?
- Are my trading partners content with mine?
- How many of these mechanisms will each organization have to support?
- Is there a tactful way to tell my customer that his network is not secure enough to meet our standards?

Currently, standards haven't really been settled for securing applications; there are at least as many approaches as there are companies (some companies use several approaches). So, naturally, no standard schemes have been accepted for securing the integrity of Web Services. Of course, PKI, SSL, and all those standards are still appropriate. What's not settled is which of those standards and familiar technologies will be used and under what circumstances, and how.

In order to resolve these issues and reach agreement on policy and technical details, you'll need a heart-to-heart talk with multiple organizations, such as several of your distribution partners or suppliers.

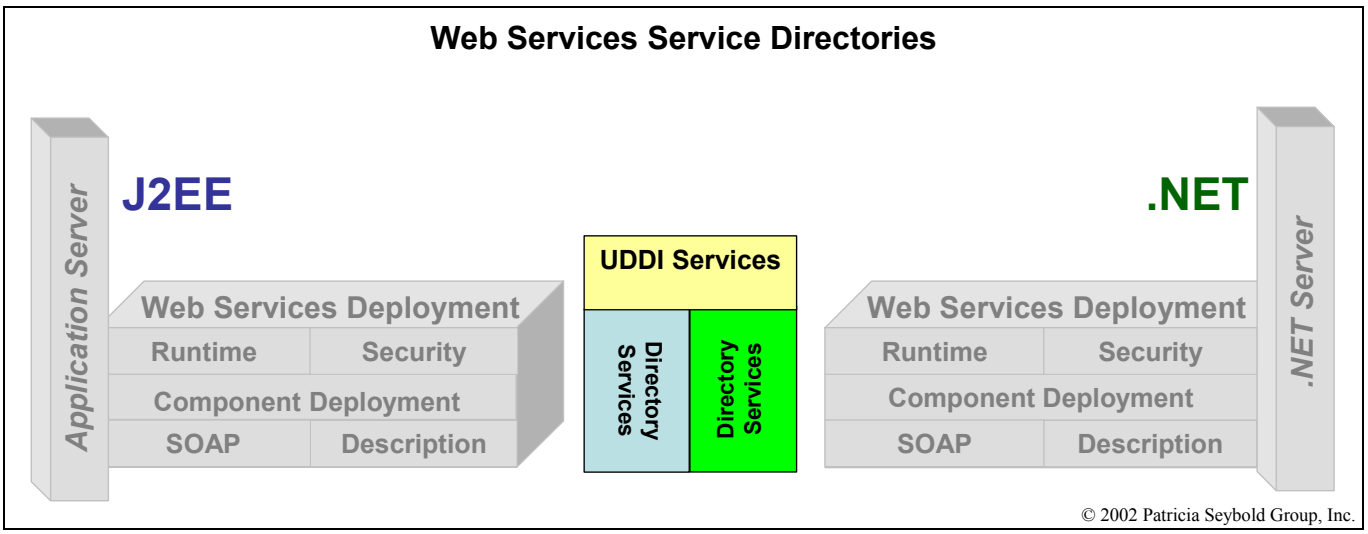

Illustration 4. In this diagram, UDDI and directory services are positioned between Web Services requestors and providers. Requestors and providers find and authenticate each other via UDDI and directories.

SERVICE INFRASTRUCTURE SERVICES

Service Infrastructure Services will ultimately cover so much ground that just about everyone will be involved in their selection. IT architects will take the lead in identifying what infrastructure services are needed, taking their cues from the line of business heads and their shared customers' scenarios. Illustration 5 highlights the position of Service Infrastructure Services.

Service Infrastructure Services represents a common set of services required by all Web Services, such as security, management, billing, state management, transaction management, and auditing, among many, many others. From the standpoint of reliability, flexibility, adaptability, and productivity, it is critical that common services be isolated from business logic rather then coded into each application, or worse, each application function. These common functions should be provided as separate Web Services.

Today, early adopters and vendors of Web Services technologies are in the process of making their lists of required infrastructure services, defining their purposes, and deciding how those services will be provided.

Some of these infrastructure services will be offered as pay-as-you-go services, much like credit card authorizations are today. More commonly, these services will be built by in-house developers, provided with other technology such as application servers and Web Services frameworks, or sold by software vendors as functional suites, to be tailored by IT developers.

A key question to ask of vendors providing the enabling software or the actual services is, "Will your services swing both ways?" In other words, will you be able to call a single service and have it manage authentication, user session management, state management, or transaction management? We're not advocating that these infrastructure service be executed identically for all types of Web Services, just that, ideally, you want to develop and manage with one set of services, rather than one for each platform. It's unlikely that vendors stuck in the I-am-the-universe mind set (Microsoft and Sun come to mind) will be offering all-embracing services. So you'll want to locate the vendors who *can* help you with this.

The purposes of Service Infrastructure Services are as follows:

- Enable developers to separate business logic from functions common to all Web Services
- Ensure that common functions, such as authentication and auditing, are, in fact, common—providing consistent service

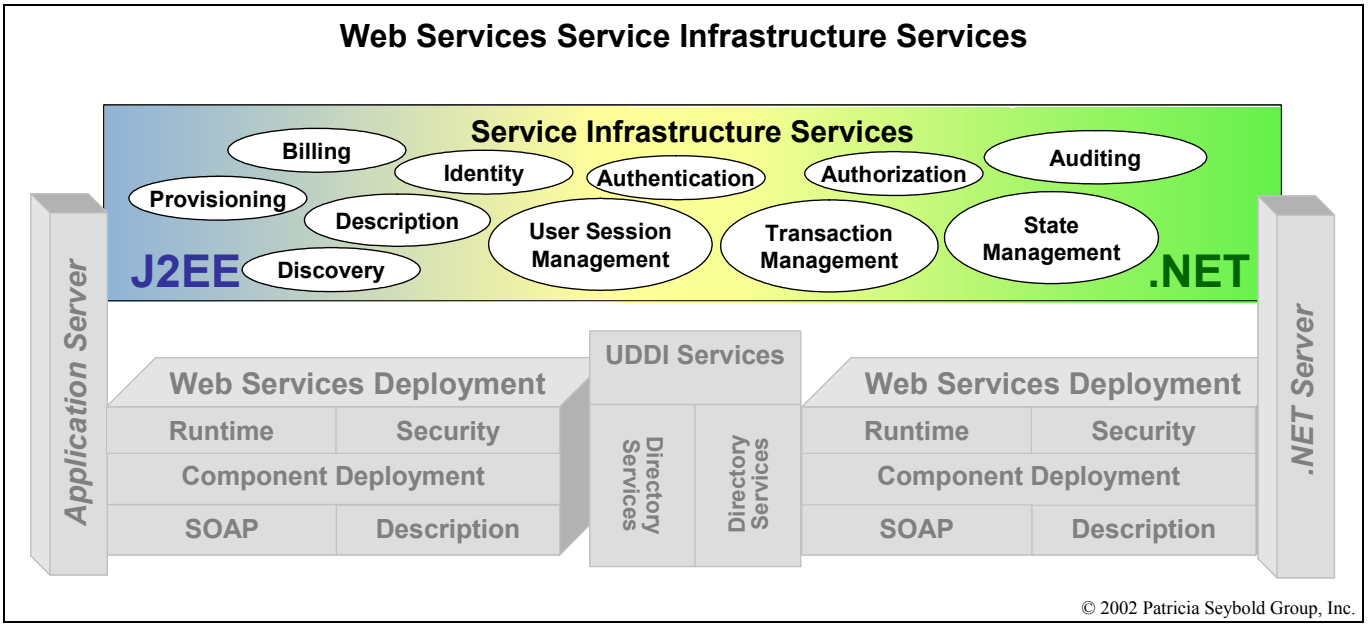

Illustration 5. Service Infrastructure Services isolate common services, such as session management and authentication from application logic. These infrastructure services might be built inhouse, provided by the application server or Web Services deployment environment, or provided by another software package. Eventually, you may be purchasing (or offering) some of these services on a pay-as-you-go basis.

- Support an architecture where new standards and technologies can seamlessly replace the old
- Enable Web Services to be instrumented so that the quality of service they provide can be monitored at all times

ADAPTIVE BUSINESS PROCESS ENVIRONMENTS

Adaptive business process environments provide the orchestration of groups of Web Services and infrastructure services into business processes.

Adaptive business process environments are, at this stage, a dream of the future, when there are so many Web Services available that, in fact, business processes can be assembled from them. There are products in this category today, but the corporate portfolios of Web Services are just now in the process of being created.

One premise of the early products in this arena is that business analysts will use these technologies to create adaptive business processes. We're skeptical of that premise, but we do foresee these technologies being used by a team comprised of business analysts, IT developers, and IT architects. These teams will be able to adapt business processes very quickly and probably enable business people to adjust business processes in narrow circumstances. For example, a purchasing agent would not be able to design an e-procurement system, but he would be able to establish a path or set of rules for orders in a new product category.

This category of Web Services technology enables the business process team to do the following:

- Specify business outcomes, business constraints, and business rules that should govern assembly
- Provide business end-user/analyst tools to modify the parameters of business rules and constraints
- Provide a mechanism for monitoring and continuously improving the performance of assembled Web Services applications and/or business processes

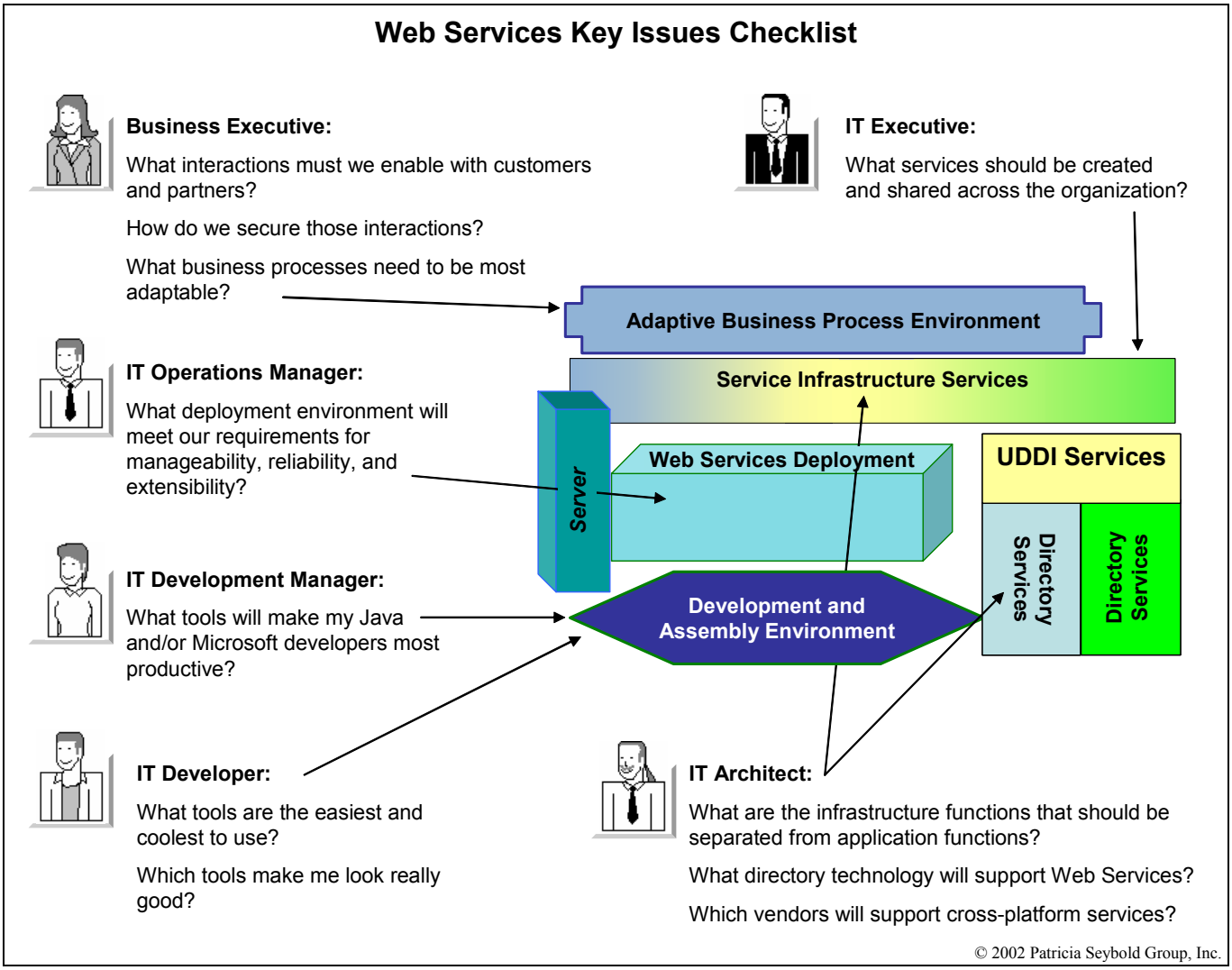

Illustration 6. There are a few key issues assigned to each of the people who are driving, using, or creating Web Services in your organization. They are depicted here in this diagram.

BOTTOM LINE

Web Services architectures are new to the IT scene, and the product offerings are immature. Nevertheless, the incipient value of Web Services is so great that IT and business leaders need to invest in understanding what they are, and developing plans for using them.

We've listed the roles of the people in your company who should be thinking about Web Services. We've laid out a map of the Web Services technology landscape. We've connected people to technology categories. Now that you've got your bearings, you're ready to lay your plans for capturing the Web Services opportunities. Your personal checklist of questions to be answered is presented in Illustration 6 as well as in worksheet format in Table A.

Patricia Seybold Group
Web Services Key Issues Worksheet

Business Executive	
What interactions must we enable with customers and partners?	
How do we secure those interactions?	
What business processes need to be most adaptable?	
IT Executive	
What Services should be created and shared across the organization?	
IT Operations Manager	
What deployment environment will meet our requirements for manageability, reliability, and extensibility?	
IT Development Manager	
What tools will make my Java and/or Microsoft developers most productive?	
IT Architect	
What are the infrastructure functions that should be separated from applications functions?	
What directory technology will support Web Services?	
Which vendors will support cross-platform services?	

Table A. Use this worksheet to assess how you are doing in preparation for Web Services.

What to Look for in a Web Services Assembly Platform

Questions to Consider as You Embrace Web Services Development

By Patricia B. Seybold

NETTING IT OUT

There's more than hype to the Web Services evolution. We're on the threshold of a more productive era in the development and deployment of applications. A new category of application assembly platform has emerged that makes it relatively easy for corporate developers to embrace many of the best practices that have been used for over 10 years by professional systems engineers in the distributed object computing field.

This new breed of application assembly environments lets developers turn components into Web Services and lets them assemble applications by combining Web Services. These Web Services assembly platforms let corporate developers use business rules and business events to determine the workflow of an application based on its context. And, because the assembly components are Web Services, each of which has already been optimized for the Net, these Web Services-assembled applications can be automatically deployed across the Internet.

But, as with any new category of products, there are key capabilities you'll want to examine carefully. The most important issues to consider are the development expertise and background of your corporate developers and the deployment environment you expect to be using. Since most businesses are embracing both Microsoft's .Net environment and the J2EE world for Web Services, ideally, the application assembly platform you select should support both environments equally well.

HOT NEW CATEGORY: WEB SERVICES DEVELOPMENT AND ASSEMBLY

We're going to see a variety of companies positioning themselves as Web Services application development environments and assembly platforms over the next 18 months. It's too early to call the winner in this space. But we can provide a few guidelines you should use in evaluating these alternatives.

A Web Services Application Development Environment is a complete Integrated Development Environment (IDE) with all the tools that a team of developers would need to design, develop, test, debug, assemble, and maintain a set of object components, Web Services, and Web Services-based applications.

A Web Services Assembly Platform assumes that developers are using some other IDE to design, develop, and test software components. A Web Services assembly platform should enable a small team of business analysts and developers to assemble Web Services from previously-created components and previously-wrapped application functionality.

We are particularly intrigued by the advent of this latter category of Web Services assembly platforms. They address a new need in the marketplace, and it's a need that is going to get much larger, not smaller: how to mix and match Web Services in order to quickly assemble robust net-Native applications.

ASSEMBLING WEB SERVICES INTO APPLICATIONS IS NOT A NEW TECHNIQUE

First, of course, you probably already know that assembling applications as collections of loosely-coupled services is not a new idea. It's the approach that savvy technology architects have been using for years.

Second, using business rules to separate out application logic from both the presentation of the application and the data it processes is also not new. The only variations on this theme we've seen of late have to do with the ease with which programmers can do their part of the process—designing and testing robust business rules—and business people can do their part—modifying business rules on the fly by changing the parametric values that trigger the rules.

Web Services address a new need in the marketplace, and it's a need that is going to get much larger, not smaller: how to mix and match Web Services in order to quickly assemble robust net-Native applications.

Third, Web Services are created in a variety of ways. Many of the capabilities that are referred to today as Web Services are simply existing application functionality (like a transaction monitor or an order entry system) with published Web Services Description Language (WSDL) interfaces that can be registered in a Universal Description Discovery and Integration (UDDI) directory, communicating via XML and Simple Object Access Protocol (SOAP).

So what's the excitement about? It's simply this: We're finally reaching a point in application assembly platforms where some of the best practices that good architects have been using for years are now being commercialized into development frameworks and platforms that mere mortals can use. As one cynical architect explained, "They keep mediocre developers from hurting themselves and the rest of the world." In fact, you'd be hard pressed to find a new application being developed commercially today that isn't developed as a collection of Web Services, usually sitting on top of a Web Services application framework. That means that extending and enhancing these newer Web Services-based applications should be easier and more straightforward than ever before.

QUESTIONS TO ASK WHEN EVALUATING WEB SERVICES ASSEMBLY PLATFORMS

If you're evaluating an application assembly framework or platform that professional and/or corporate developers might use in-house to develop new applications such as Bowstreet's Business Web Factory, SilverStream's eXtend, or BEA's "Cajun" (WebGain Studio follow-on product), there are a number of important questions you'll want to ask.

These same questions are probably also useful ones to ask when you're evaluating an off-the-shelf packaged application like PeopleSoft 8 or mySAP that is purported to be designed as a set of customizable and extensible Web Services, or when you're assessing an application infrastructure, like BEA's WebLogic or IBM's WebSphere.

- Who are the target customers/users of the platform? Professional developers (J2EE, CORBA), corporate developers, (Visual Basic, Scripting languages, Business Rules), and/or business analysts (business process flow and parameters for business rules)?

- Which development languages and tools should the target customer/user be comfortable with—Java and VisualAge for Java or Forte or VB and VisualStudio.NET?

- Is the runtime environment for the assembled applications primarily J2EE or .Net? Or is it truly ecumenical? (Generally, these environments are either J2EE or .Net-centric but accepting of services created in the other environment.)

- How robust, scalable, and performing are the applications that result from using this environment in a standard manner? (Any application can be tuned for performance—but you need to know how apps will run if "run of the mill" developers work in this environment.)

- How are business rules created and maintained? Who creates them? How are they implemented? How are they tested and staged? Who can modify the parameters? Is there a clear separation of the business logic that business users can alter (e.g., the parameters) and the application logic that developers need to write, test, integrate, and maintain?

- How well does the environment support loose vs. tight coupling? In other words, does a requesting service need to maintain the connection in order to receive a response from another service? Tight coupling can become problematic in a networked world in which intermittent connectivity is the norm. But many applications will need to support both loose and tight couplings.

- How is metadata defined and how accessible is it? What tools exist to enable developers and analysts to adhere to and/or to modify and extend the metadata definitions that are published as part of Web Services?

- How are Web Services managed and re-used?

As the Web Services arena evolves, more questions will arise. In particular, we expect to see a lot of debate and development around application metadata, metadata repositories, and semantics. Today's Web Services assembly tools will only really work for Web Services that are developed by sets of people who agree on terminology and semantics. (When I issue the request, "check price," which price do I get—the published list price, my company's discounted price, or a dealer's price?) As soon as we start trying to link in third-party-created Web Services, the possibilities for ambiguity become so great as to make those linkages unusable. But, as higher level standards emerge for XML schemas and business process metamodels, it will eventually become feasible to assemble applications out of both known and unknown-but-trusted Web Services.

WHERE'S THE BIGGEST OPPORTUNITY?

We see a huge opportunity that is being neglected by most of the players in the burgeoning Web Services assembly arena. What most corporate development shops need is an agnostic platform that will accommodate .Net-created Web Services and J2EE-created Web Services in a relatively seamless and painless manner. The reality in today's corporate world is that most IT and application development and assembly shops use both J2EE and Microsoft .Net. The major advance that Web Services offer is the opportunity to assemble applications out of both J2EE and .Net-created Web Services. Although Web Services standards promise interoperability, you'll find that most of today's players and platforms have a strong pro-Microsoft or pro-J2EE bias. So, as you're investigating alternatives, check to see how well the Web Services Platform you're considering will address the needs of both groups of developers.

We hope that the Web Services industry responds to this wake-up call soon!

What's Next? If you find this report valuable, then:
Print It – YES, you are free to print and freely distribute this report as long as its contents are not changed.
Send It – Send a friend or colleague to http://www.psgroup.com/vm/ws/ so they can download their own copy.
Stay Informed – Subscribe to our free e-mail newsletter at http://www.psgroup.com/signup.asp to stay up-to-date on this and other important research topics.
Contact Us – Contact us at feedback@psgroup.com to find out how our additional research and consulting services can help your organization sort through its Web Services strategy.