



Patricia Seybold Group

Trusted Advisors to Customer-Centric Executives

The Evolution of Service-Oriented Architecture

From Integration to Business Scenario
Development

*By Brenda M. Michelson
Contributing Architect*

UNAUTHORIZED REDISTRIBUTION OF THIS REPORT IS A VIOLATION OF COPYRIGHT LAW

210 Commercial Street, Boston, MA 02109 • Phone 617.742.5200 • Fax 617.742.1028 • www.psgroup.com



The Evolution of Service-Oriented Architecture

From Integration to Business Scenario Development

By Brenda M. Michelson, Contributing Architect

January 6, 2005

SERVICE-ORIENTED EVOLUTION

Topping the list of must-have technology strategies in 2005-2006 is service-oriented architecture (SOA). Services and SOAs are an important IT architectural strategy that will change how software is developed, used, and sold.

However, SOA is not new; leading companies have been employing services architectures for years to provide consistent multichannel experiences to their customers,¹ to integrate with partners, and to reap IT efficiencies through reuse.

What is new is what you can do with SOA. We believe the current uses of SOA (integration, adding application functionality, building new service-oriented applications) are merely the first stages of the service-oriented evolution. In the longer term, SOA will be the springboard that propels IT organizations away from traditional application development toward delivering IT instantiations of business scenarios, or business scenario development.

In business scenario development, IT business solutions will be compositions of services, business events, and business processes matching the interactions of your business—with customers, partners, employees, and regulatory agencies—in the support of commerce, collaboration, and information exchange.

While the idea of business scenario development is intriguing, in order to get there, you need to start at the beginning. You need to establish solid service-oriented practices, a deep service catalog, and an extensible SOA environment.

¹ See our case study, “L.L. Bean’s Architecture Evolution,” September 9, 2004, <http://www.psgroup.com/doc/products/2004/9/CS9-9-04CC/CS9-9-04CC.asp>.

IN THIS REPORT

In this report, we share our view of the service-oriented evolution from two perspectives. The first is, “What can you do with services and SOA?” To answer this question, we present the evolution as a series of business scenarios, using our fictional customer-adaptive retailer, Digits Undercover. The second perspective is, “What should you be thinking about to adopt (or evolve) an SOA?” Here we build on our earlier services and service-oriented architecture work and share our evolved thinking about services and the SOA environment.

Let’s start with our definitions of service and service-oriented architecture.

OF SERVICES AND SOA

What Is a Service?

Simply stated, a service is a thing that fulfills a purpose. A service is, in essence, a “worker,” employed to achieve a specific end goal for a requester. The end goal may be small in scope, such as retrieving information, or large in scope, such as executing a business process. Most services are in the middle, completing a function. The scope of a service is referred to as its grain, or level of granularity.

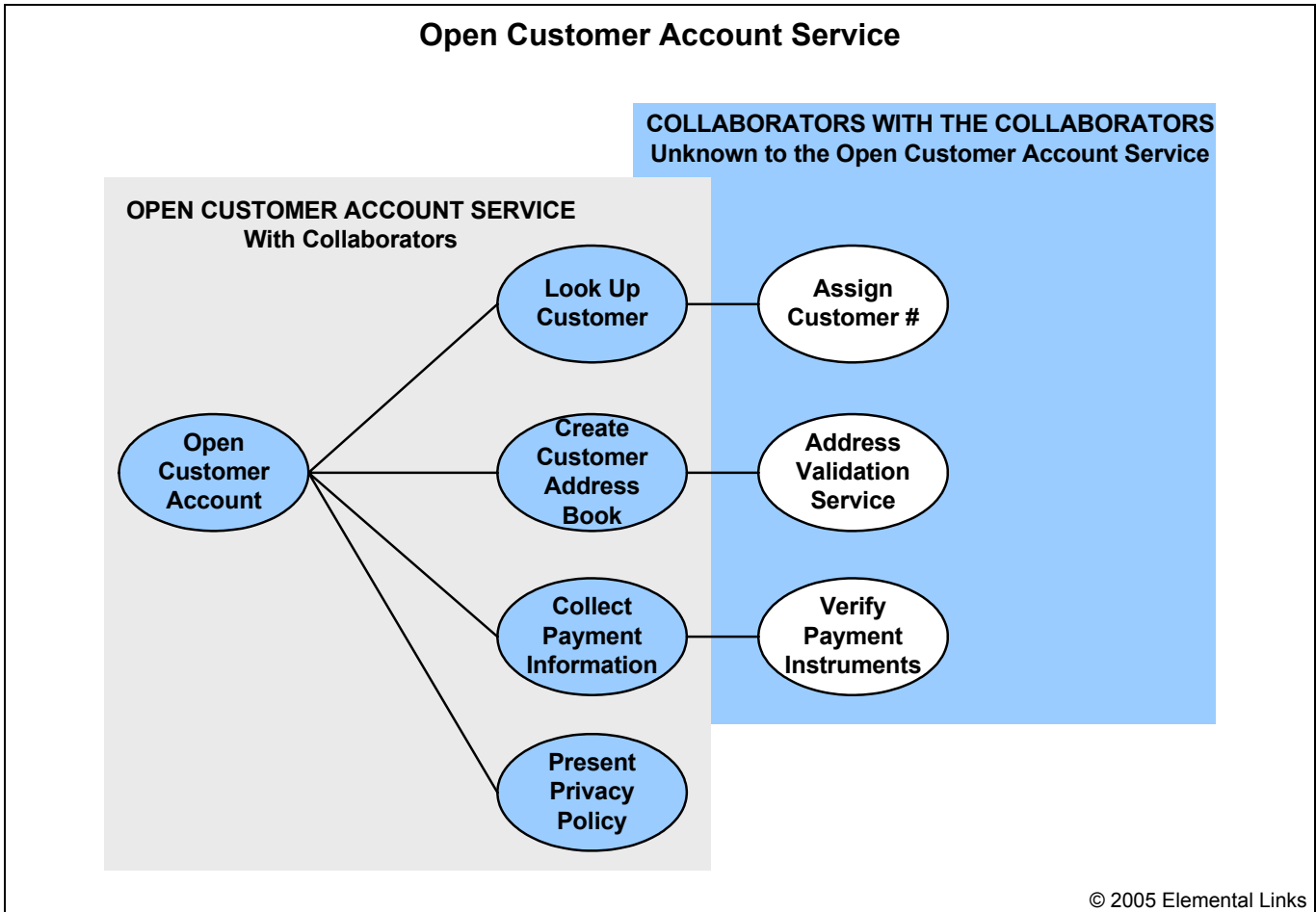


Illustration 1. This diagram shows how services call on other services to fulfill their obligations. This allows for services to be larger grained, fulfilling a broader purpose, such as completing a business process, without breaking the rules of service definition.

WHAT KIND OF THING IS A SERVICE? A service is an abstract resource that has a name, a job, job tasks, contact information, and policies regarding security and service levels. The job tasks of the service correspond to physical code assets (objects, components, legacy code, application package APIs), but only the service knows these specifics. From the requester’s point of view, a service is a (small) black box. To use (request) a service, you send a message—in accordance with the contact information and policies—and then (if appropriate) receive a reply message. The details of the service execution are unimportant to you. You are only concerned that the service does its job and returns understandable results.

A SERVICE’S JOB. The job of a service is limited to a single distinct business concept, function, or process. This characteristic is referred to as the bounds of a service. Finding the correct bounds is a key factor in service definition and building your collection (catalog) of services. A service may call upon other services if it needs assistance in completing its job. This service-to-service relationship is called collaboration. Illustration 1 shows an example of services collaborating to perform all of the required tasks to open a customer account. These same services may be used independently (Look Up Customer) or in other collaborations (Address Validation in a Create Customer Address Book service).

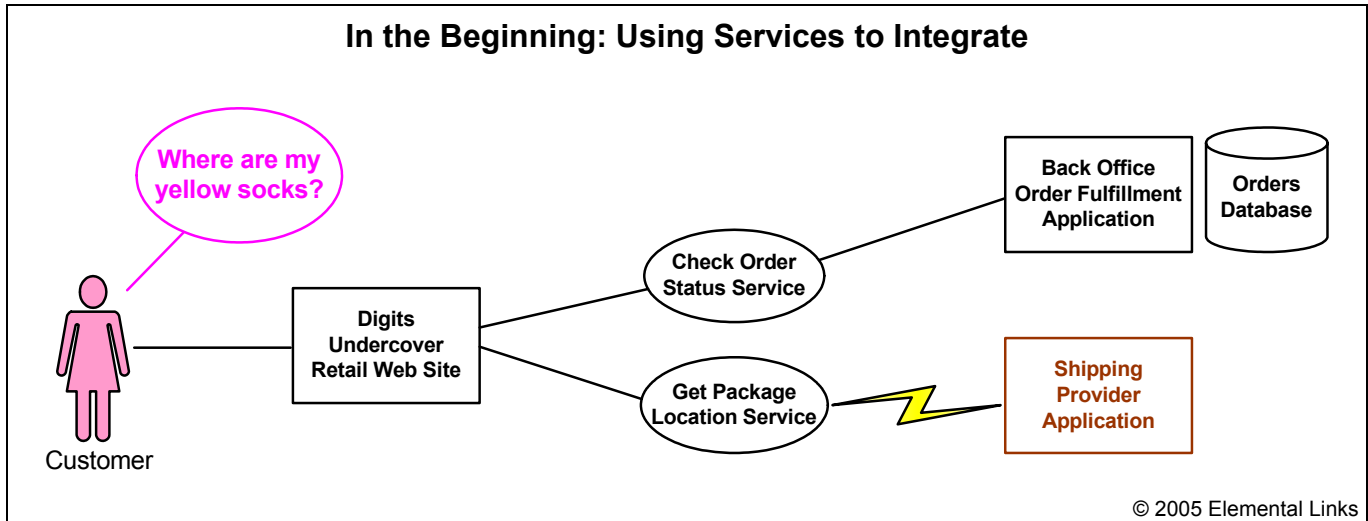


Illustration 2. This is a common usage of SOA, creating new services to extend back-end and partner located information to your customer Web site. In this example, our customer wants to know if her yellow sock order has shipped and, if so, when she can expect it.

What Is an SOA?

AN ARCHITECTURAL CONCEPT. Architecturally, SOA refers to the loose coupling of a service to its provider. As we discussed earlier, a service is really an abstract resource. This concept is not limited to functional assets (services). You can apply it to larger IT services such as network, platform, or even programming services. If a requester knows what a service offers (job, service levels) and how to use it (contact, security), then it really is not important (to the requester) how that service works, as long as the results meet expectations. The implementation details of the service are the responsibility of the provider. We refer to this as “service-oriented thinking.” We find this thinking helpful in planning and executing large initiatives (such as architectural adoption), software product bundling, and defining organizational structure and responsibilities.

AN ENVIRONMENT. SOA also refers to the collective environment that allows services to be defined, developed, and used by other services, and to be assembled into business solutions by adding process, user interface, and business rules. Beyond service development and business solution assembly, the SOA environment provides for service discovery, policy definition and enforcement, quality of service (performance, availability, reliability, and load),

transaction compensation (undo), and usage metering.

WHAT CAN YOU DO WITH SERVICES AND SOA?

In this section, we take an evolutionary view of service usage in SOA. We start with the use of services for simple integration, and we work our way to services as the foundation for business scenario development. The evolution as presented here also makes for a good SOA adoption plan.

Integrate

Often, an IT department’s first use of services is for integration. Services can bring back-end functionality to the customer, provide multichannel consistency, and tap into resources locked in a legacy system or residing with a partner. In Illustration 2, we show a classic service integration example of a customer going to a retailer’s Web site to check the status of her order. Employing services in this manner benefits both the customer and the retailer. The customer is able to use self-service to check her order status (particularly important in the gift-giving season), and the retailer benefits by providing a low cost, yet accurate, customer service mechanism.

Looking at the services in our example, the Check Order Status service asks a back-end order

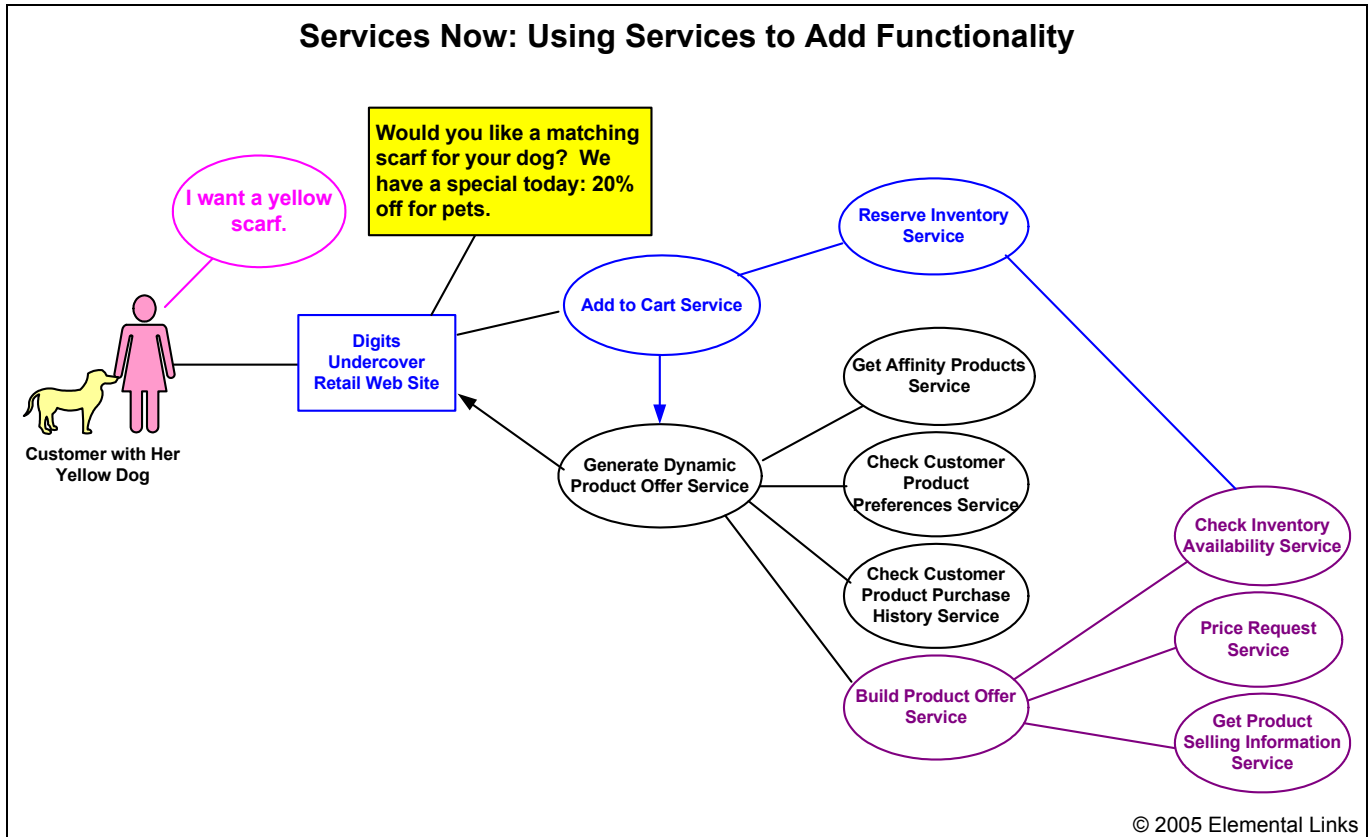


Illustration 3. SOA is becoming increasingly popular as a means to add new functionality to existing applications and to create new applications. In this example, our customer receives a promotional offer after adding a yellow scarf to her shopping cart.

fulfillment application to retrieve the order status and returns (if applicable) the package tracking information. The order status and package tracking information is displayed on our retailer's Web site. If the customer chooses to, she may click-through on the package-tracking link to retrieve up-to-date package location and arrival information direct from the shipper's system (using the Get Package Location service).

It is important to note we have defined our services, Check Order Status and Get Package Location, in accordance with what they do, not how they do it or how they are implemented. This is a basic rule of service definition. You should always define a service in accordance with its purpose, expressed in business terms, even in integration scenarios. Had we defined the Get Package Location service as Get Package Location from Brown, and then decided to change shippers or add a new one, we would have had to change not only the Get Package Location

from Brown service, but also all of the requesting applications. Instead, we defined our service according to its job, getting package location information regardless of the end provider of the service (Brown, Blue, or Green). Defining your services well (purpose, bounds, granularity) is the key to building a reusable service catalog. This keeps you positioned for replacement and/or extensibility.

Using services for integration is a great way to achieve some early SOA benefits, such as reuse, consistency, and customer empowerment.

Add Functionality; Build Service-Oriented Applications

Moving beyond simple integration, you can use services to add new functionality to existing applications and as the basis for new applications. In Illustration 3, we use services to conduct ecommerce (adding to cart, checking and reserving inventory,

up-selling, and building a product offer). In our example, the customer adds a yellow scarf to her cart, and, based on our retailer's affinity products and the customer's preferences and purchase history, we generate a dynamic offer, or up-sell. In this example, our services call on other services (collaborators) to accomplish their work. The Generate Dynamic Product Offer service collaborates with four services to complete its work. Even the collaborators have collaborators, as shown by the Build Product Offer service. The Check Inventory Availability service participates in multiple collaborations (Reserve Inventory and Build Product Offer). We could also use the Check Inventory Availability service directly.

As you employ services to add functionality and build new applications, you are delivering solutions to your business while simultaneously enlarging your service catalog. Suddenly you have a collection (or catalog) of services that you can mix and match to fulfill multiple Customer Scenarios[®] across channels and lines of business. Your customers benefit from a consistent, accurate experience, and you benefit from reuse and agility. Once you create the Generate Dynamic Product Offer service, you can deploy it on other sales channels. If you decide to change the business rules for generating a dynamic offer, you only have to do that in one place.

Another great thing to do with services is to support an application or portfolio transition strategy. You can deliver new functionality to your existing applications (as services) and then use those same services in the new application. This allows you to build for the future and still deliver new functionality to your business. In the example above, our retailer could be developing a new service-oriented ecommerce application to support its growing business. In the interim, it decides to roll out the new Generate Dynamic Offer service to its existing Web site to gauge customer response and fine-tune the offer algorithms. When its new ecommerce solution is ready, it will use the same (or finer-tuned) Generate Dynamic Offer service.

Be Event Driven

A powerful thing to do with services is to combine them with events, to be event driven. We define an event as something notable that happens inside or outside your business. An event (business or system)

may signify a problem or impending problem, an opportunity, a threshold, or a deviation. The occurrence of the event prompts an action, either human or automated.

An automated action may include the invocation of one or many services. As well as being the target of an event-driven action, services can be the creators of events. You can instrument your services to generate events based on notable business or system conditions.

Illustration 4 shows a business scenario in which a high-value customer places an order for a red sweater, which ends up on backorder. Our retailer has decided that if a high-value customer (based on lifetime value) has an item go on backorder, the customer will receive a \$10 gift card. Our retailer's customer advocate may choose to contact this customer if retention is at risk.

The backorder scenario starts with the customer checking out. The Order Checkout service sends the order to the Order Fulfillment service. For our retailer, a backorder is a notable deviation, which results in the generation of a Backorder Event. The Backorder Monitor service sees this event, determines the backorder is in fact for a high-value customer, and notifies the retailer's campaign management system and the customer advocate that a high-value customer has a backorder.

Combining services and events allows you to break out of the traditional (limiting) boundaries of applications. You can easily connect a customer, partner, or employee action with back-end processes—across your enterprise and even out to your partners. In the example above, our retailer turned a potentially bad customer service experience into a positive one by connecting order fulfillment (operations) with customer retention (marketing).

Our retailer could also use events to manage its inventory better—generating a Low Inventory Threshold event when a customer's inventory reservation brings inventory below a desired threshold. This inventory threshold event could prompt an inventory reordering process. Now our retailer can balance customer service and inventory carrying costs.

When discussing events, we always issue two standard warnings: The first is that if you define everything as an event (a notable thing), then truly nothing is an event. You will have a clogged event pipe-

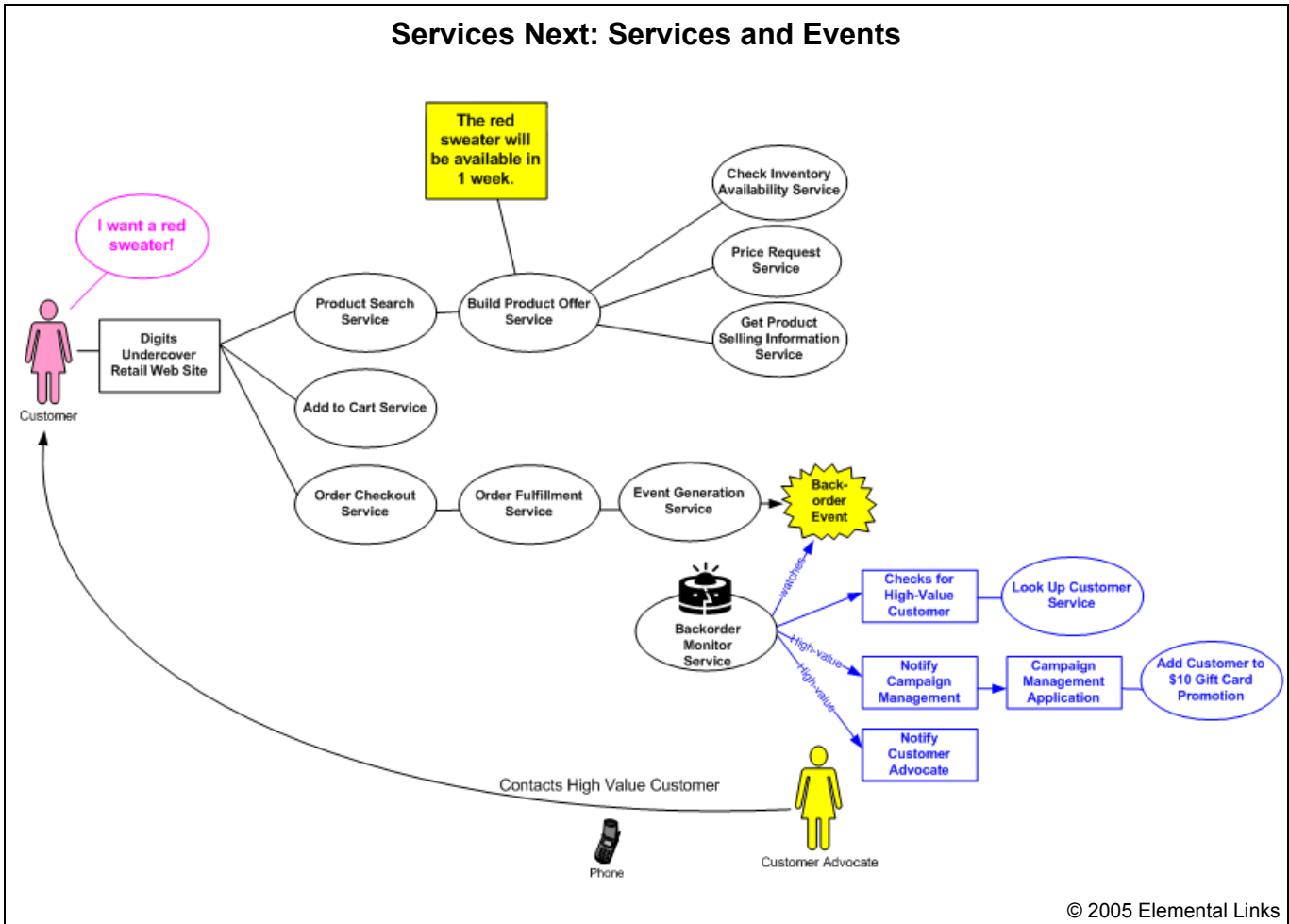


Illustration 4. Services and events are a natural (and powerful) combination. In this example, a high-value customer places an order, which goes on backorder. As part of our retailer’s retention strategy, it issues a \$10 gift card to high-value customers whose orders go on backorder. In this business, a backorder is a deviation from the norm and is a notable event.

line.² The second warning is that you must define your events in business terms. The name of the event and information it contains must be meaningful to your business, not just to your applications and databases.

² Please refer to our first Business-Driven Architecture (BDA) report, “Creating a Blended Architectural Portfolio,” October 14, 2004, <http://www.psgroup.com/doc/products/2004/10/BDA10-14-04CC/BDA10-14-04CC.asp>, for a discussion of using an event pattern for notable enterprise events, type-specific event systems (stock quotes, build-to-order), and information dissemination (all orders to a data warehouse).

Execute and Participate in a Business Process

Services are starting to emerge in business process automation (a.k.a. process-based architecture). Services play two roles here: First, the job (or purpose) of your service may be to execute a business process. Second, services may be included in or invoked by the steps of a business process.

In Illustration 5, we show a customer requesting a special order from our retailer. The customer wants socks that match his company’s color scheme and carry his company’s logo. He needs the socks for a sales conference in four weeks. In a special order situation, the customer wants a price quote with de-

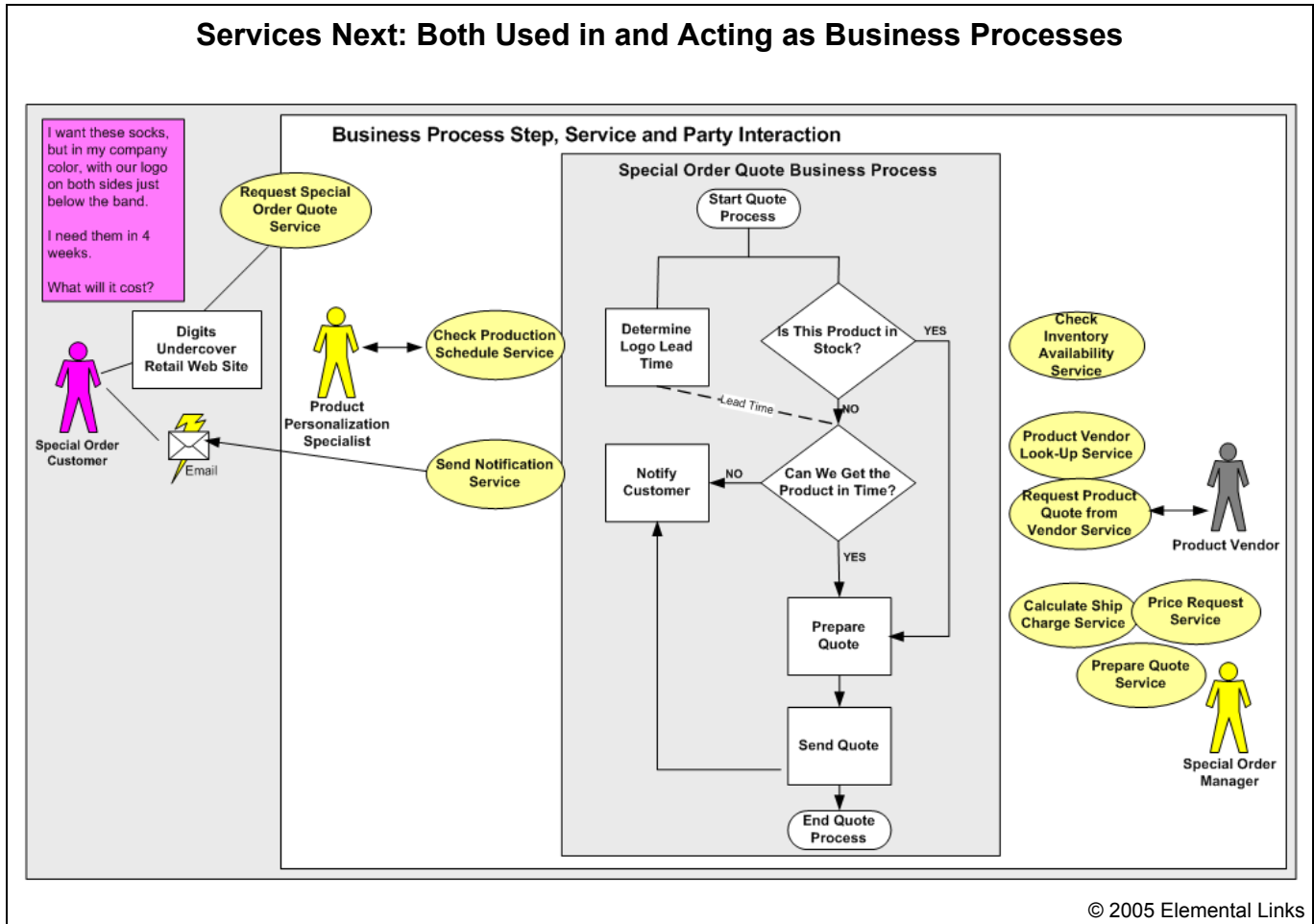


Illustration 5. Services are starting to be used to execute business processes and/or to fulfill the work of the business process steps. In this example, our customer is requesting a quote for a special order. Our retailer needs to check with the sock personalization area, current inventory, and possibly the product vendor to prepare a quote for the customer.

livery date. For our retailer, the quote process includes checking with the sock personalization department, reviewing current inventory levels, and possibly contacting the sock product vendor. After all this information is collected, the retailer's special order manager can prepare and send a quote to the customer.

Using services, we have created a Special Order Quote service responsible for controlling the entire quote process. The steps of the process invoke services (reusing Check Inventory Availability and Price Request) to perform the work. Our customer Web site initiates the business process through the Request Special Order Quote service. We could easily expose this same business process to other sales

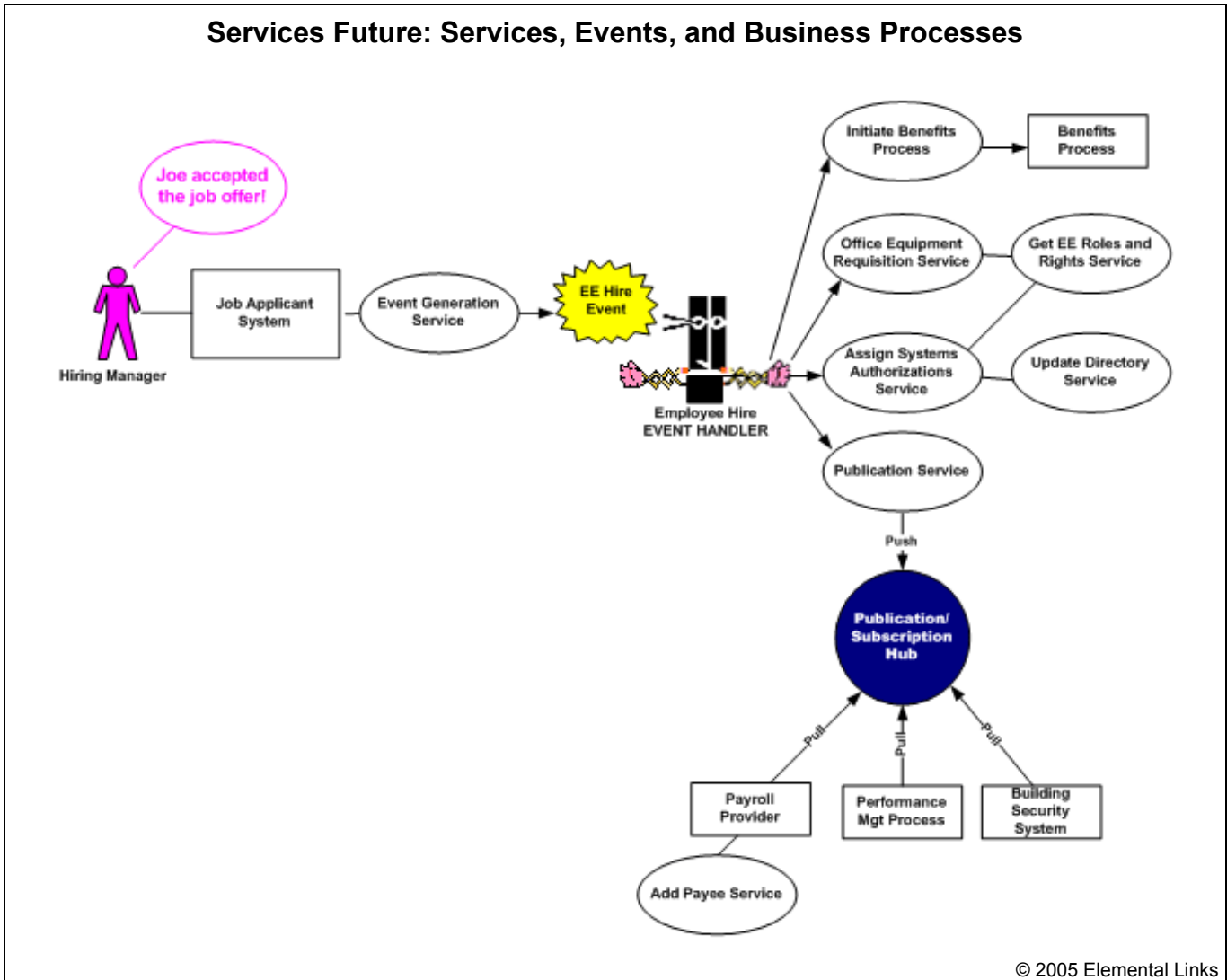
channels, increasing our company's presence while ensuring a consistent experience with our customers.

While business process oriented services offer great potential, it is still early in their technology lifecycle, with standards stabilizing and products emerging.³

Business Scenario Development

We have discussed the relationships between services and events and services and business proc-

³ See our first BDA report for more information on process-based architecture, "Creating a Blended Architecture Portfolio," <http://www.psgroup.com/doc/products/2004/10/BDA10-14-04CC/BDA10-14-04CC.asp>.



© 2005 Elemental Links

Illustration 6. Services are the springboard for business scenario development. In this example, our hiring manager marks a new employee, Joe, as hired and then goes to lunch. While our hiring manager is gone, the entire process for bringing a new employee on board is executed, including the scheduling of Joe’s benefits eligibility in 30 days and his six-month performance review.

esses. The next evolutionary step is to combine (or blend⁴) the three. When you can compose business

⁴ In a blended architecture, you choose from a portfolio of architecture strategies, or blends. You can select the right architecture strategy for each business situation, and, more importantly, you can seamlessly combine the architecture strategies within the course of a business interaction. For more information on blended architecture, see our first (mentioned earlier) and second (“Designing a Fluid Enterprise Using an Adaptive, Customer-Centric IT Architecture,” December 9, 2004,

solutions from services, events, and business processes, your IT solutions correspond to the true nature of your business. At this point, you are moving away from traditional application development to business scenario development.

While there a multitude of customer-driven scenarios that you could implement in this manner, we are showing an internal scenario to demonstrate the coverage provided by SOA. In Illustration 6, we

<http://www.psgroup.com/doc/products/2004/12/BDA12-9-04CC/BDA12-9-04CC.asp>) BDA reports.

handle the often labor-intensive process of bringing a new employee on board. The scenario starts with the hiring manager marking one of his job candidates, Joe, as hired in the job applicant system. The job applicant system creates an Employee Hire event as our hiring manager goes off to lunch. The Employee Hire event triggers all the downstream activities (benefits, office equipment, system security, building security, add to payroll, and performance management) to be completed before Joe starts at our retailer.

In this example, we use services to complete event-driven actions—one of which is a business process (Initiate Benefits Process). We also use system services for event generation and publication.

It is easy to imagine how many of your business scenarios can be achieved by mixing and matching services, events, and business processes. It all starts with SOA—and so should you.

Advantages of a Services Approach

Using our simple examples, you can see that SOA offers both business and IT advantages.

Business advantages consist of the following:

- **Consistent Experience.** An SOA can provide a consistent experience for customers and partners across channels and lines of business.
- **Business Agility.** An SOA can add new functionality, expose functionality to new channels, and vary functionality based on context (customer, partner, entry point).
- **Mix and Match.** An SOA can compose business solutions from a reusable service collection, leveraging internal and external services.
- **Optimize Interactions.** An SOA can optimize business interactions for customers, partners, and internal constituents through the implementation of business scenarios versus traditional applications.

IT advantages include:

- **Reduction of Costs.** Reuse of services reduces IT development and maintenance time and costs.⁵
- **Leverage Existing IT Investments.** Your service providers are existing code (objects, components, legacy modules, and application package APIs) and information assets (databases, files, and documents).
- **Transition Strategies.** An SOA can provide application and portfolio transition strategies.

EXPANDING OUR NOTION OF SERVICES

In discussing the evolution of SOA, we presented some new ideas on how services are used and how they behave. To realize the potential of services and SOAs, we need to expand our notion of services and SOAs. In this section, we share “Release 2” of our service classification scheme and revisit the truths of services and SOA.

All Services Are Not Alike!

In our examples, we have all kinds of services. Some of our services perform discrete tasks (Check Inventory Availability), others perform complex functions (Generate Dynamic Product Offer), and some are business processes (Special Order Quote Process). There are services that do system jobs (Event Generation Service), services that serve the enterprise (Price Request), services that are limited to a domain (Add Payee), and even services that monitor conditions for us.

SERVICE CLASSIFICATION SCHEME. It is with this diversity in mind that we developed a five-aspect service classification scheme. (See Table A.) The five aspects are service type, service class, service level, service composition, and service interaction. The service classification scheme is a great tool to help you think about your services during service

⁵ Note that the ROI from reuse typically occurs between the second and third use of the service. This note is based on industry metrics that consider the increased design and development time to “get the service right” to be reusable.

Service Classification Scheme		
Aspect	Instance	Description
Type	Service type describes the primary behavior or work pattern of the service.	
	Request/Reply	The Request/Reply is an information-bearing service. The service either retrieves information and/or performs a calculation/manipulation on behalf of the requester to produce a result. The Request/Reply may perform information update tasks, but it often calls upon a worker to add, change, or delete information.
	Worker	The Worker performs a function, most likely resulting in a change of state of the thing being worked on. The Worker may return a “done” message.
	Monitor	A Monitor is a service whose job it is to observe something and report on its findings based on monitoring and notification rules.
	Agent	An Agent is a service whose job it is to observe something and then act on its findings. An Agent shares behavior with a Monitor in that it observes based on monitoring rules, however an Agent differs from a Monitor in that it may take action(s) based on its findings.
	Intermediary	An Intermediary is a service that intercepts a service request (or reply), performs a value-added function (usually translation or policy), and then forwards the enhanced message to the original target.
	Aggregator	The Aggregator is a service that combines results from federated sources or services. A Request/Reply service may use the Aggregator. Aggregator services will play a role in right-time architecture, business activity monitoring (BAM), and the Grid.
	Process	A process is a long-running service, coordinating the actions of multiple parties through a series of work steps. The work steps may be services.
Class	Service class describes the functional areas served by the service.	
	Business	The service provides functions that map to the business architecture.
	System	The service provides functions that map to the technical architecture.
Level	Level is the scope served by the service, in respect to both the function provided and the typical constituency.	
	Domain	The service provides function specific to one area and is generally consumed by a narrow set of requesters. Domain Service examples include Generate Invoice, Pay Employee, and Receive Goods.
	Enterprise	The service provides a function that crosses multiple domains and/or fulfills a purpose for a multitude of requesters. Enterprise service examples include Generate Notification, Value Inventory On Hand, and Identify Customer.

Service Classification Scheme (continued)		
Aspect	Instance	Description
Level (continued)	Global	The service, often externally provided, represents a consolidated view of a thing from multiple parties and may be used by any requester. Global Service examples include Check Credit Rating Service and Global Directory Look-Up.
Composition	Composition is the basic structure of the service.	
	Base	A Base services is an independent service that alone fulfills a purpose. A Base service may be included in a composite.
	Composite	In a Composite service, two or more services collaborate to fulfill a purpose. The participating services may be either Base or Composite services.
Interaction	Interaction describes how the services collaborate to fulfill the purpose.	
	Orchestration	An Orchestration is a type of collaboration in which the primary service directly invokes other services. The primary service knows the sequence of actions and the interfaces, responses, and return states of the called services.
	Business Interaction	A Business Interaction is a type of collaboration in which some coordination mechanism knows the sequence of actions, possible states, and paths of interaction between one or more services. The Business Interaction is usually long lived and involves requests/messages from multiple parties. The coordination mechanism may be a Workflow, Business Process Manager, or Event Handler.
	Interception	An Interception is a type of collaboration in which an intermediary service receives and acts on a request (or reply) and then forwards the request (or reply) to the original target. In many interception scenarios, the requesting and providing parties are unaware of the collaborating service. Interception is used to perform common functions such as security, policy, audit, and translation, in a non-invasive manner.

Table A. This table describes our service classification scheme.

discovery⁶ and service definition. The classification scheme assists you in determining if something is a service (service type), finding the correct boundaries (service type, class, level, composition), planning security requirements and service levels (service class, level), and surfacing requirements for your SOA environment (service interaction).

⁶ The Patricia Seybold Group has a Service Discovery and Definition Methodology that is an extension of Customer Scenario[®] Mapping.

Truths about Services and SOA

Some of the “truths” have changed, such as:

- Services may be stateful, and, as discussed above, not all services behave the same. New Web Services standards,⁷ Grid Services stan-

⁷ Web Services standards related to stateful services include: WS-Addressing, WS-AtomicTransaction, WS-BusinessActivity, and WS-Coordination. See <http://www-106.ibm.com/developerworks/library/specification/ws-tx/#ba> and <http://www.w3.org/Submission/ws-addressing>.

dards,⁸ and business process notation and execution languages⁹ are driving the emergence of stateful services.

- Services may include a user interface. Using a technology such as Web Services for Remote Portlets (WSRP),¹⁰ services will be able to serve mini-applications to requesters, in their portals. This will be important for business performance dashboards and universal task boxes.

Some new “truths” are:

- Services will start playing a role in analytic applications. Aggregator services will gather results in business intelligence queries¹¹ and will gather events and instrumentation information from the execution of your business scenarios.
- Service-oriented architecture is not just for business solutions. Services and SOA constructs are used to deliver architecture and infrastructure solutions (event generation, event handling, subscription, and business process execution).
- Services should monitor and analyze the health of other services and the IT instantiation of the business scenarios. Services should include instrumentation for IT and, as applicable, business performance. Business scenario and architecture operations monitor services should be in place to receive and review the instrumentation output.
- Services are starting to appear in IT shops that have “buy over build” strategies. Services are now available from external providers, in limited

⁸ See OGSi Grid Services Standards (http://www-unix.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf) and Web Services Resource Framework (<http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf>).

⁹ For more information on business process execution language (BPEL) and business process management notation (BPMN), see our first BDA report.

¹⁰ See http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp.

¹¹ Grid Services and Grid Data technologies will enable this shift. We describe our vision for Grid technologies in the enterprise in our first BDA report.

amounts from software companies, and more frequently from the true service providers, such as financial institutions, credit card providers, and the Amazons and Googles of the world. As you shop for services, look for functionality, ease of use, service levels, security, fail-over, usage metering, and consolidated billing.

Other “truths” still hold, including:

- Services do not have to be Web Services, but they can be.
- You cannot predict all of your potential services.
- You cannot predict all the possible uses of your services.
- If your applications directly consume numerous fine-grained services, you will end up with a fat user interface or complicated—and repetitive—application controller logic. Leverage composition when possible.
- Services and SOAs are not a silver bullet. If your data, applications, infrastructure, and/or business processes are broken (redundant sources, conflicting definitions, bad fit for the business problem, or inefficient), fix them.
- Ownership and organization issues can sink an SOA. Prepare your organization for reuse.¹² Validate your business drivers for SOA.

SOA FOUNDATION

In order to have a successful SOA strategy, one that supports initial SOA uses (integration, adding new functionality, service-oriented application development) and prepares your organization for the

¹² The architectural premise of SOA (service is an abstract resource, defined in business terms, loosely coupled with its provider) sets SOA apart from previous strategies (objects and components) that promised reuse. Services are available to any requester (and can come from any provider) that can send, receive, and interpret messages. Services (through composition) ease the burden of the requester by offering larger-grained functionality, targeting business functions and processes.

future (business scenario development), you need to build a solid foundation.

The foundation starts with the services mindset—understanding what services are, what they can do for your business, and how to “think services.” Once you have established your services mindset, you need to work on the SOA cornerstones, your service catalog, and an extensible SOA environment.

In this section, we share key requirements to jump-start your thinking on both.

Services Catalog

Your services catalog starts with good service fundamentals, including:

- Define your services in accordance with best practices of naming, purpose, boundaries, and granularity.¹³
- Use a common business lexicon so your services can speak to each other, your applications, and your business scenarios in known and agreed-upon business terms.
- Design in (directly or through collaborators) systems management, security, service usage and business performance instrumentation, error handling, compensation (undo), and logging.

Support your services catalog with a repository to help you accomplish the following:

- Use the repository to advertise your services to business solution builders. Include service metadata such as name, purpose, classification, disposition, current uses, and service levels.
- Use the repository to feed your runtime environment: registry, policy, service level, and state (on/off).
- Leverage the repository to plan and analyze service performance. Keep service level metric targets. Relate repository metadata to runtime data

¹³ For more information on services and granularity, see “Defining Services at the Right Level of Granularity,” by Robert E. Shelton and Patricia B. Seybold, August 15, 2003, <http://www.psgroup.com/doc/products/2003/8/GRANULARITY8-03/GRANULARITY8-03.asp>.

from service instrumentation, usage, and logging data.

SOA Environment Requirements

Develop your SOA environment with extensibility in mind, as follows:

- Make it easy for your business solution builders to assemble services into applications and business scenarios.
- Provide the ability for services to collaborate. Think forward to improvisational collaboration based on requester context.
- Allow for both static and dynamic runtime service discovery.
- Allow for the non-invasive placement of intermediary or interceptor services to perform common functions: security, audit, and/or translation.
- Consider and implement security and policy enforcement mechanisms at the onset.
- Provide for transaction failures. Implement mechanisms and procedures for service coordination, fail-over, and transaction compensation (undo).
- Know what is going on your SOA environment. Implement systems management for your services and infrastructure. Advertise and measure quality of service (performance, availability, reliability, and load), and service usage.

SUMMARY AND NEXT STEPS

In sharing our view of the services evolution, we brought to light new opportunities for services and SOAs, along with the associated implications for your services mindset and SOA foundation.

As demonstrated by our examples, as your service catalog grows, so do the possibilities for your business. To get your service catalog started (or fortified), in a forthcoming report, we will share our Service Discovery and Definition Methodology—which is an extension Customer Scenario[®] Mapping.

The Service Discovery Methodology leverages concepts presented in this report, including our service classification scheme. Applying this methodol-

ogy, you can find and define the right services to fulfill your customer- and partner-adaptive business scenarios.