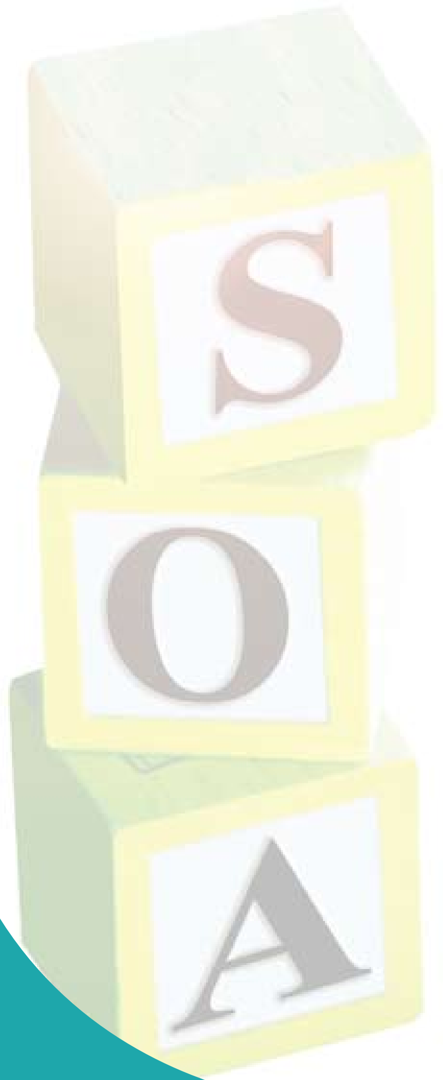
Integration and SOA

Concepts, Technologies, and Best Practices

Beth Gold-Bernstein & Gary So

ebizQ webMethods®



About ebizQ

ebizQ – The Insiders Guide to SOA and Integration

ebizQ is the leading community of business and IT executives, managers, architects and developers focused on business integration and SOA. ebizQ's analyst-centric multimedia content includes interactive training, webinars, and analysis of industry trends and best practices. You can access daily aggregation of in-house and industry-wide blogs, breaking news, feature articles and white papers.

Learn more by visiting www.ebizQ.net and gain access to the following great content!

Webinars: www.ebizQ.net/webinars/
White Papers: www.ebizQ.net/white_papers/
Online Curriculum: www.ebizQ.net/trainingcenter
Blogs: www.ebizQ.net/blogs
CIO audio: www.ebizQ.net/executive_corner/cioaudio/
Buyers Guide: www.ebizQ.net/vendors

Bring the rest of your team up to speed. Attend our online companion course, a Manager's Guide to Integration and SOA: www.ebizQ.net/managersguide or call
1-866-55-EBIZQ (1-866-55-32497)

About webMethods

Get There Faster.

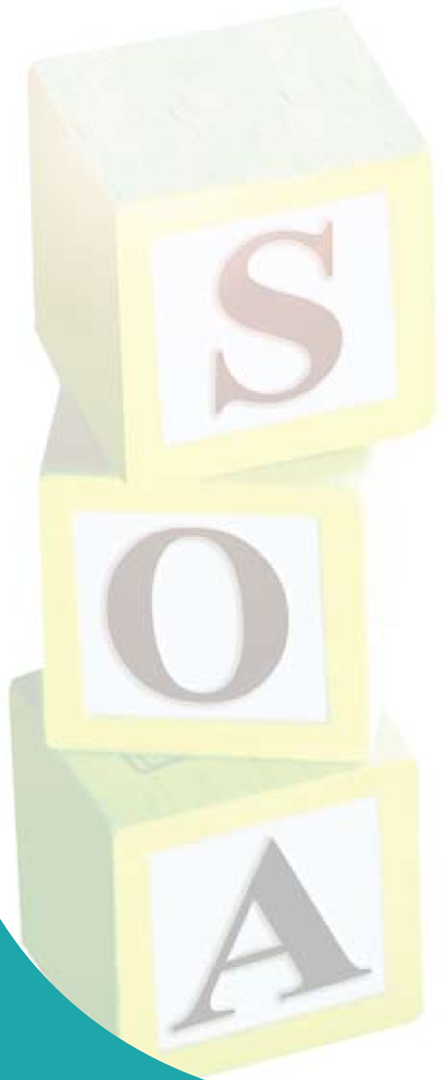
webMethods provides total business integration to the world's largest corporations and government agencies. webMethods flagship product suite, webMethods Fabric, leverages leading SOA and BPM technologies and methodologies to deliver rapid ROI to our 1,300 customers around the globe. With webMethods, customers can take a process-centric approach to their business problems, allowing them to re-use their existing IT assets, dramatically improve business process productivity and ROI, and rapidly create competitive advantage by making their business processes work harder for their company. webMethods (NASDAQ: WEBM) is headquartered in Fairfax, VA, USA and has offices throughout the United States, Europe, Asia Pacific and Japan.

For more information on our company and our products, we invite you to visit our website, www.webMethods.com

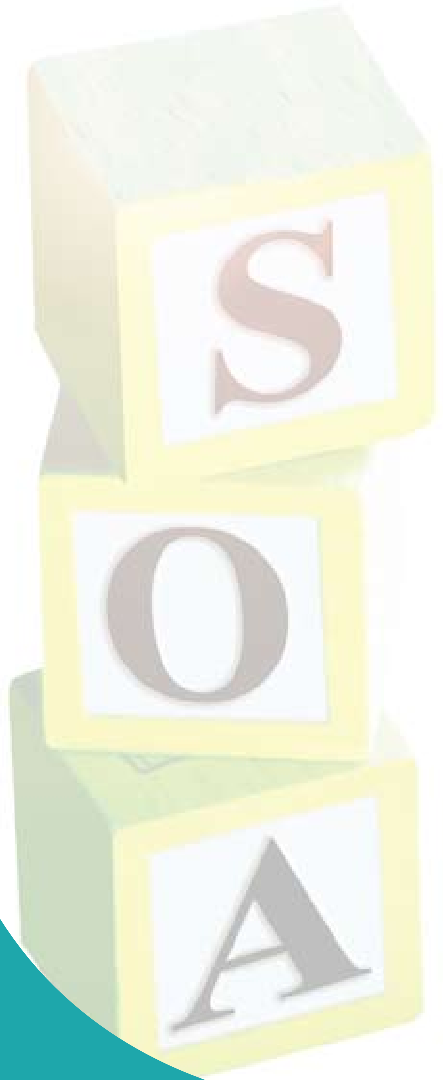
View our SOA webinar series – www.webMethods.com/Events/Web/GetRealwithSOA
Learn more about us – www.webMethods.com/About
Hear our vision – www.webMethods.com/Vision
Meet our customers – www.webMethods.com/Customers
View our products – www.webMethods.com/Products

If you would like additional copies of the book, please email soabook@webMethods.com

Tables of Contents



5 Integration and SOA



Executive Overview

Service Oriented Architecture (SOA) is transforming IT, so it is important to understand the relationship between integration and SOA. This is not a straightforward task, however, because definitions and interpretations of SOA abound.

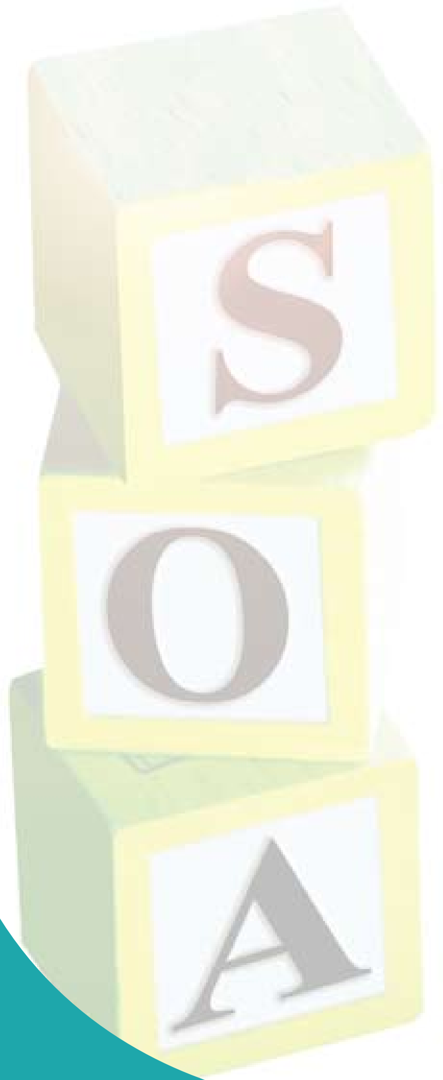
From a software engineering perspective, SOA is simply an approach to software design that involves assembling systems from reusable components (services) that may originate from different sources and underlying technology environments. At this level, SOA is often associated with Web services. However, Web services address only the low-level interactions between services; they do not address the grander architectural and design philosophies of SOA. In other words, SOA is about much more than just Web services.

From a broader IT perspective, SOA suggests a fundamental shift in how an organization implements business systems—with attendant changes in technology, methodology, and organizational structure. An even broader interpretation is that SOA marks the end of monolithic enterprise applications and the beginning of more flexible and adaptable business process-centric applications.

While definitions of SOA vary, there is unanimous agreement on SOA's value and promise. The past decade has seen unprecedented leaps in computing. More and more aspects of a business use computers. Additionally, the Internet has extended system interfaces to include connections with trading partners. Not surprisingly, this has resulted in an unruly explosion of stovepipe applications, databases, and electronic connections. The challenge for IT today is to create and support an infrastructure that can harness these diverse assets and deliver on new business requirements more effectively. SOA provides the basis for this infrastructure and offers the promise of tangible business benefits. SOA also makes it possible for organizations to leverage IT assets so they can rapidly deploy new business solutions as needed.

Chapter 5

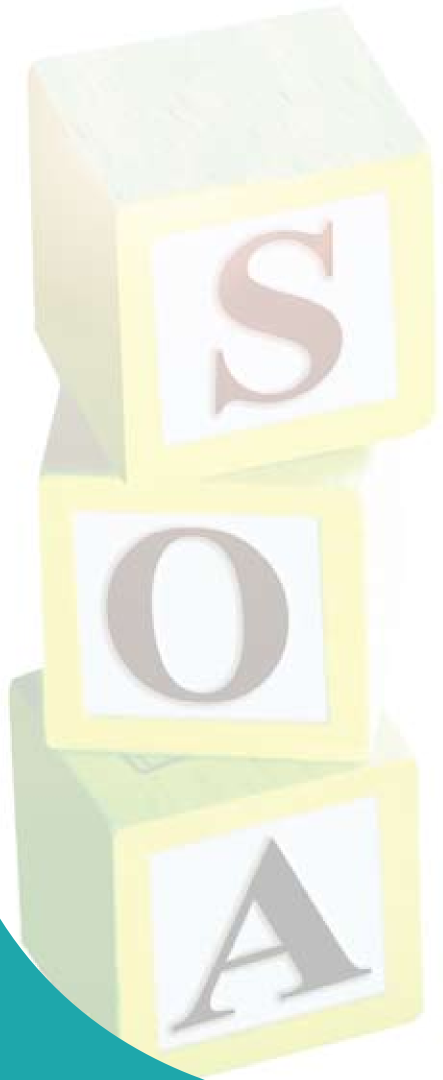
Integration and SOA



With its promise of transforming the way systems are developed, SOA will benefit how organizations approach integration. At a basic level, the growing adoption of Web services will improve system-to-system interoperability—making it simpler to connect systems—while the componentization of business services will make it easier to create new business solutions from existing applications and retrieve information as the business demands. At the same time, integration technologies play an important role in the SOA portfolio, addressing needs such as service orchestration, service- and process-level monitoring, service management, and the ability to make legacy systems part of the SOA environment. Clearly, integration and SOA are highly complementary, with SOA making it easier to integrate systems and business processes, and integration technologies providing the essential technical infrastructure services required for SOA implementations.

Unlike previous IT revolutions that were accompanied by massive change, SOA enables an evolutionary approach and the opportunity for organizations to move forward incrementally. The key to success is to plan strategically at an enterprise level and then implement tactically, with the appropriate best practices and governance policies and procedures in place. This strategy delivers benefits along the way, making it easier to justify SOA investments.

This chapter explores the benefits, design concepts, technologies, and best practices of SOA, and shows how they relate to integration.



Case Study 5-1
Company: WRA Electronics

Company Description

WRA Electronics is a pseudonym for a Fortune 500 global provider of electronic components and computer products. The company serves as a supply channel partner for more than 500 suppliers and 150,000 original equipment manufacturers, contract manufacturers, and commercial customers. Headquartered in New York, WRA employs more than 10,000 people in more than 200 locations around the world.

Business Challenges

As a player in the global electronic equipment market, WRA is required to comply with RoHS, a European industry directive restricting the use of six hazardous substances in electrical and electronic equipment. As part of the directive, WRA needed to make all relevant information about its products accessible to its customers for quoting, ordering, and purchasing decisions. However, different business processes (e.g., quoting and ordering) employed different subsystems running on different platforms and diverse technologies, and the global components data was located in an Oracle database.

How Integration Addressed the Challenges

WRA adopted an SOA strategy to address its requirements. Using webMethods as its SOA platform, WRA was able to link its disparate systems using Web services and to provide a service-oriented interface for consumer applications. The webMethods solution also provides connectivity to WRA's mainframe-based systems that currently cannot interoperate with Web services. The RoHS Web service provides data from the Oracle database and plugs into the existing business processes for quoting and ordering. Any differences between the platforms are bridged using the Web service interface, and the webMethods platform performs all necessary mappings and transformations.

Bottom-Line Business Benefits

WRA is now able to comply with the RoHS mandate. The new system provides global enterprise visibility into the company's products and parts along with RoHS information. Each month, more than 500,000 transactions pass through the company's internal Web services integration environment, and the RoHS Web service is utilized more than 80,000 times.

In addition to complying with the directive, WRA has achieved significant cost savings and efficiency improvements with its SOA approach. The system has enabled the automation of numerous process steps, saving time and errors. Company sales personnel have also been able to help customers make more informed purchasing decisions, increasing customer satisfaction. By providing its customers with quick and easy access to RoHS-related information, WRA has gained a competitive edge in the market.

Why SOA?

SOA provides many benefits to both IT and the orga-nization at large. The most important benefit—and the one most often cited in relation to SOA—is reusability. SOA allows organizations to do more with the resources at their disposal. It promotes reusability by enabling services to be shared by different applications running on different platforms. Reusability creates a competitive advantage by providing greater flexibility in the way computer systems can be used to support the business.

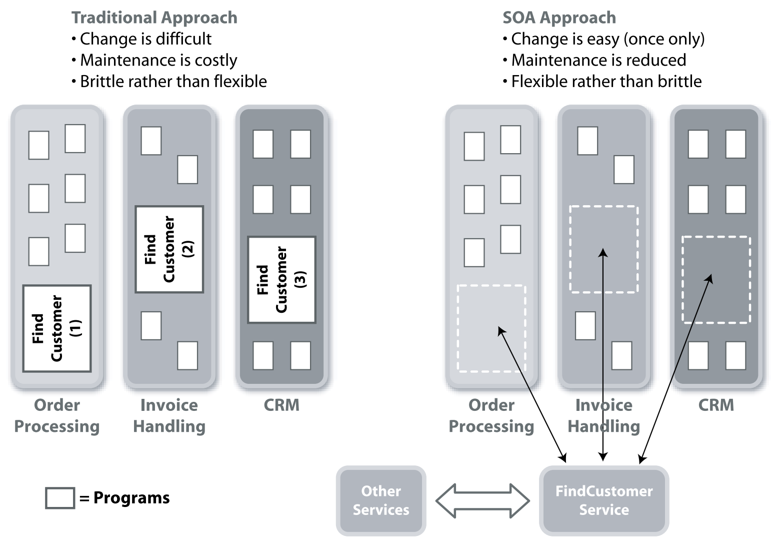
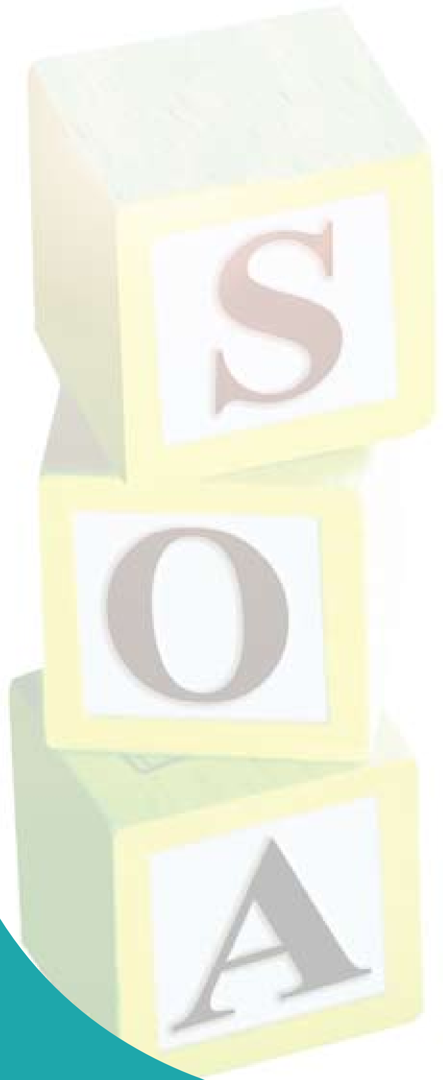


Figure 5-1. Flexibility of SOA through reuse

For example, many organizations have stovepipe applications that address specific areas of business functionality, such as order processing, invoice handling, or customer relationship management (CRM). Each application has its own customer data and its own view of the customer. And each has local mechanisms for retrieving information, such as customer details, that were developed in the environment and technology specific to the application platform. In an SOA, these duplicated routines can be re-

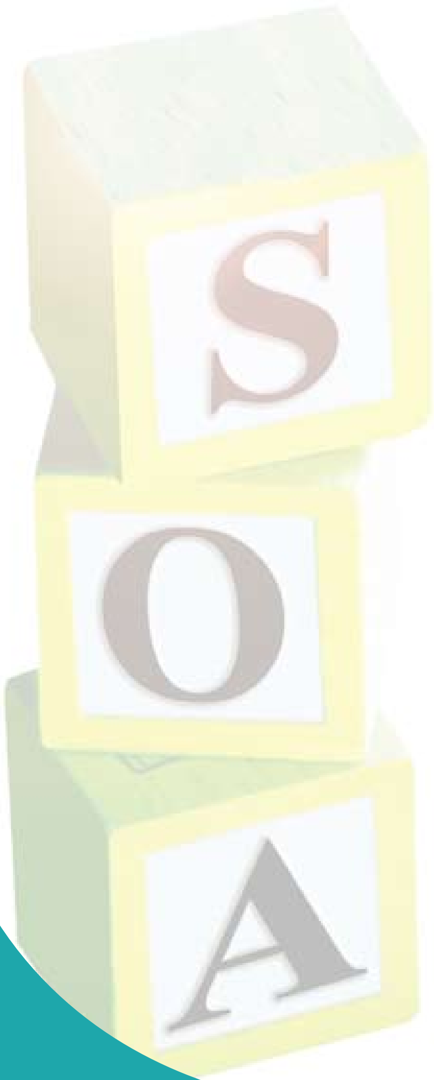


placed by a common reusable service (for example, FindCustomer) that delivers the same functionality. Because there is just one copy of the FindCustomer service, changes need only be made in one place. This reduces application maintenance costs and enables new changes to be accommodated more quickly and easily.

Another benefit of SOA is increased agility when addressing business opportunities or changes in business requirements. By reusing common steps and developing only the new steps that are required, organizations can more easily modify or enhance business processes and applications when the need arises. For example, a company that offers home loans may have a loan approval process consisting of numerous SOA services. If the company now wants to offer car loans, it can reuse many components from the existing implementation, such as checking current account details or market interest rates. This increased agility translates directly into increased competitiveness. Thus, organizations that master SOA will gain an advantage over those that do not.

A third benefit of SOA is the reduced complexity from a more orderly enterprise architecture and development approach. Modular services enable a less-complex, divide-and-conquer approach to software development. This enables SOA to accelerate the deployment of new application functionality, decreases the testing effort by making it easier to test individual components independently, and lowers implementation risks through the reuse of functionality that is already quality-tested and proven.

Another benefit of SOA is increased IT adaptability. Packaged application implementations, custom-built solutions, and system changes resulting from mergers and acquisitions can be integrated more quickly and easily. SOA's inherent platform-independent approach provides organizations with more flexibility to use the software and hardware of their choice. This allows them to engage in a multisource strategy and reduces the threat of vendor lock-in. For solutions developed in-house, it enables programmers to use their existing technical skill sets.



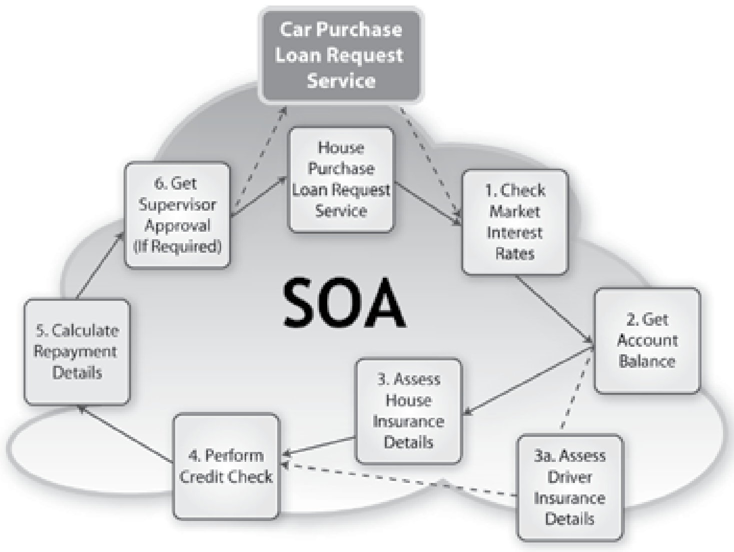
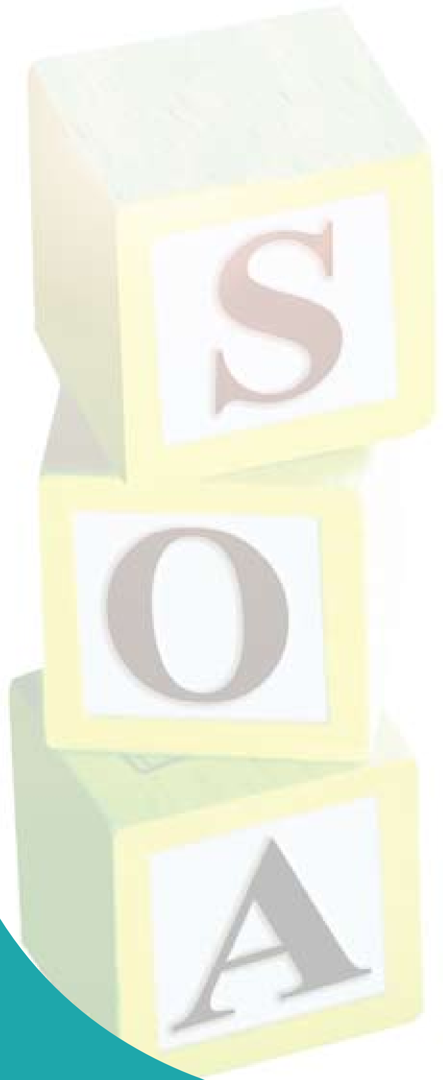

Car Purchase Loan Request Service
House Purchase Loan Request Service
1. Check Market Interest Rates
2. Get Account Balance
3. Assess House Insurance Details
3a. Assess Driver Insurance Details
4. Perform Credit Check
5. Calculate Repayment Details
6. Get Supervisor Approval (If Required)
SOA

Figure 5-2. Adding new functionality easily

Best of all, these benefits can be realized quickly. SOA is ideally suited for incremental deployment, with investments made step-by-step for individual projects. Returns can also be realized incrementally. This eliminates the need for costly and risky "big-bang" software implementations, reduces investment risk, and delivers value more quickly. Thus, SOA becomes a far more palatable investment for organizational decision makers.

SOA Explained

Although the term Service Oriented Architecture first emerged in the 1990s, the spirit of SOA existed long before that. The underlying principles of SOA—componentization, encapsulation, separation of interface from implementation, loose coupling, and so on—have been promoted since computing's infancy. These design principles were used in the days of centralized computing, when the focus was on IT resource optimization.



Then the landscape changed. On the technology side, the arrival of distributed computing offered great productivity and price performance, and the emergence of the Internet promised ubiquitous connectivity. On the business side, IT was increasingly seen as something that could be used for business advantage rather than just back-office operations. But these changes introduced considerable challenges in linking together different technologies and platforms to utilize the increasingly diverse IT infrastructure to meet new business needs. The advent of object- oriented computing paved the way for SOA by introducing the concept of having discrete program components with standard inputs and outputs that could be used to assemble applications.

So what exactly is Service Oriented Architecture? From a software engineering standpoint, SOA is an IT architecture based on the delivery of reusable, well-defined business services underpinned by IT components in such a way that the providers and the consumers of the business services are loosely coupled and have no knowledge of one another's technologies, platforms, or locations.

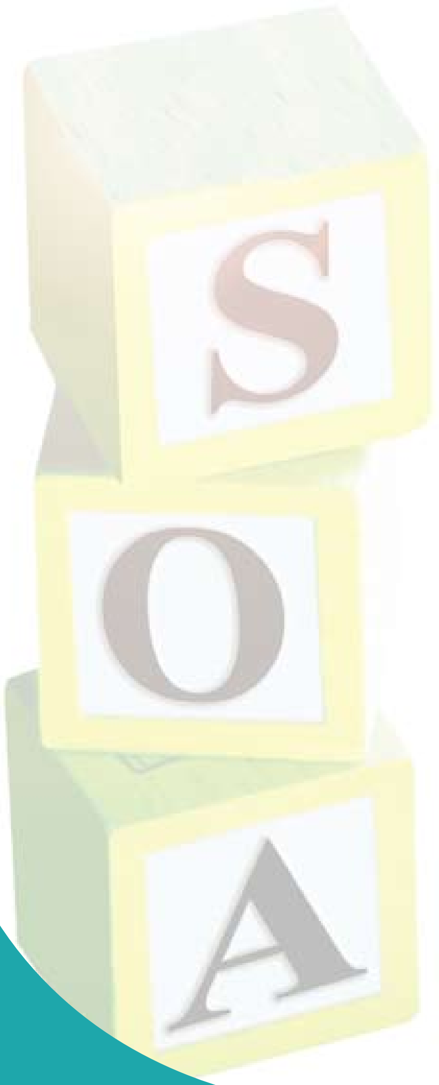
An explanation of the term "business services" is helpful here. With SOA, IT components are formed into services that are relevant to a particular business operation. For example, the service for FindCustomer might involve code that locates the customer number by searching for the customer's name in one system, then more code that uses the customer number to query the customer's service subscriptions in another application, and yet more code to retrieve the organization's interaction history with the customer from a third database.

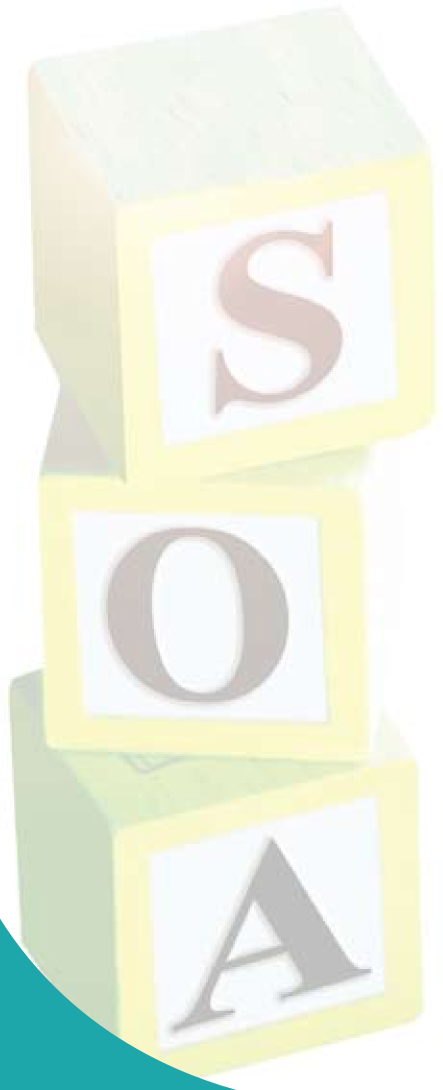
A service consists of a well-defined service interface and the service implementation. The service interface describes how to call the service. It specifies, among other things, the location of the service and the information required to make a request to the service and get a response. This is where Web services provide an ideal fit for SOA. The use of Web services ensures a standard protocol for communication between the service consumer and provider and a standard way to define a service's interface. It is basically an electronic description of how to call the service from other programs.

The service implementation is the actual code that fulfills the functionality of the service. It is the logic that resides on computers somewhere on the network and executes when the service is called, subject to appropriate security constraints. The service implementation is inherently platform-dependent. However, SOA is not concerned with how services are implemented. Architecturally, it does not matter, for example, whether the FindCustomer service is written in Java, C#, or COBOL. The only thing that matters is that the service fulfills the behavior specified by its interface.

Another important term used when defining SOA is "loose coupling." Services that are loosely coupled can be invoked by any application on any platform in any location. All the calling application needs to know is how to invoke the service, and any inputs and outputs required. This enables the interoperation of different applications and services written in different languages with different development tools and platforms. Loose coupling also means that one service can be replaced by another—provided it has the same signature (inputs and outputs)—without impacting any of the other connected services or requiring them to be modified. That means incremental changes to a system can be made without having to redevelop the entire application. The dynamic nature of SOA is another important distinction from more traditional application development approaches.

A final consideration when thinking about SOA is "granularity." Granularity refers to the level of functionality provided by a service. A fine-grained service (such as VerifyZipCode) performs a very specific and discrete function, while a course-grained service (such as ProcessInvoice) incorporates more functionality and typically maps more closely to the notion of a business service. This distinction sets SOA apart from previous computing paradigms by providing a common language for IT and the line of business to discuss computing requirements. Rather than talking about low-level bits of programming code, SOA allows application requirements to be expressed in terms of modular, reusable business services. At a technical level, SOA enables services to be made up of other services, which makes it possible to assemble course-grained services that map to useful business functions. An important goal when designing services is to encapsulate business functionality at an appropriate level of granularity so it can be reused across different solutions.





The Role of Web Services

As mentioned, SOA is often discussed in conjunction with Web services, but the two are not synonymous. In theory, SOA does not require Web services, and simply implementing Web services does not result in an SOA. However, Web services are the first standard for service-oriented computing to gain the near-universal support of software vendors, end-user organizations, and standards organizations. By standardizing essential elements of SOA, Web services remove the risk of having to bet on a particular technology (as was the case with CORBA in the days of distributed computing). As a result, Web services and their related standards are driving the widespread adoption of SOA.

The Web services standards define XML-based protocols and interface descriptions that enable services to interact. Basic Web services standards include Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI).

XML, or Extensible Markup Language, is a World Wide Web Consortium (W3C)-recommended standard for defining different types of data. XML provides a text-based way to describe and apply a tree-based structure to information. It has several advantages over previous approaches to describing data: It is readable by humans and machines, it is platform-independent, and it has the flexibility to handle arbitrary types of data and data structures. Propelled by the adoption of the Internet and open standards, XML has emerged as the critical standard for data definition and document markup.

SOAP is a lightweight protocol intended for exchanging XML-based messages in a decentralized, distributed environment, normally using HyperText Transfer Protocol (HTTP), the protocol used between Web servers and browsers on the Web. HTTP was chosen as the primary application-layer protocol for SOAP because it works well with today's Internet infrastructure and is firewall friendly—unlike other distributed computing protocols, which are generally filtered out by firewalls.

A SOAP message consists of a one-way transmission from a SOAP sender to a SOAP receiver. Applications can use SOAP to create more complex interaction patterns, including request/response and multiple, back-and-forth (conversational) exchanges. The most common pattern is the remote procedure call (RPC) or request/reply pattern, where one program (the client) sends a request message to another program (the server), and the server immediately sends a response message to the client.

WSDL is an XML-based format for describing Web services. WSDL defines the public interface to a Web service, which describes how to communicate with the Web service as well as the operations and messages supported by the Web service. WSDL is typically used in combination with SOAP. A client program reads a Web service's WSDL to determine what functions are available on the server, and then uses SOAP to actually call one of the functions listed in the WSDL.

UDDI is an XML-based standard for a Web services registry (i.e., directory) that allows service providers to publish information about their services and enables potential service consumers to browse listings of available services (using SOAP messages) to identify the ones most suited to their application requirements. As such, UDDI is often described as a Yellow Pages for Web services. Originally intended as a public resource, UDDI is now more commonly used within organizations as a private registry of their internal Web services.

Additional Web services standards are being developed in areas such as security, reliable message delivery, and transactions. Standards are also being created to define the business process flow across services (for example, BPEL, which was discussed in Chapter 4). Unfortunately, as history has taught us, standards alone do not guarantee interoperability. Vendors may implement subsets of the standards for expediency, or supersets of the standards for enhanced functionality, but these subsets and supersets may actually limit interoperability.

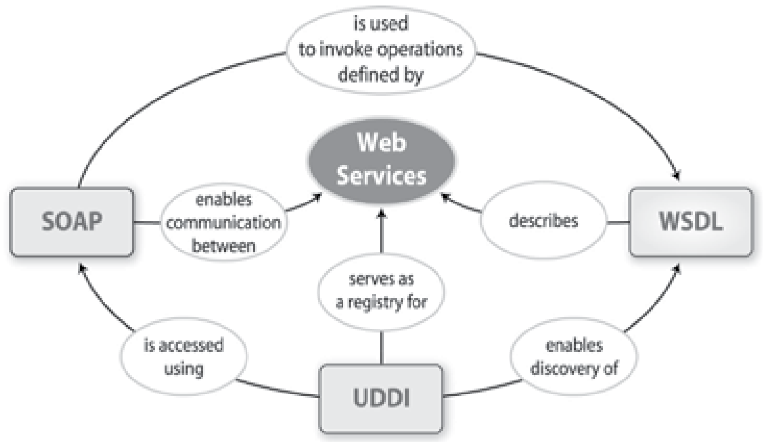
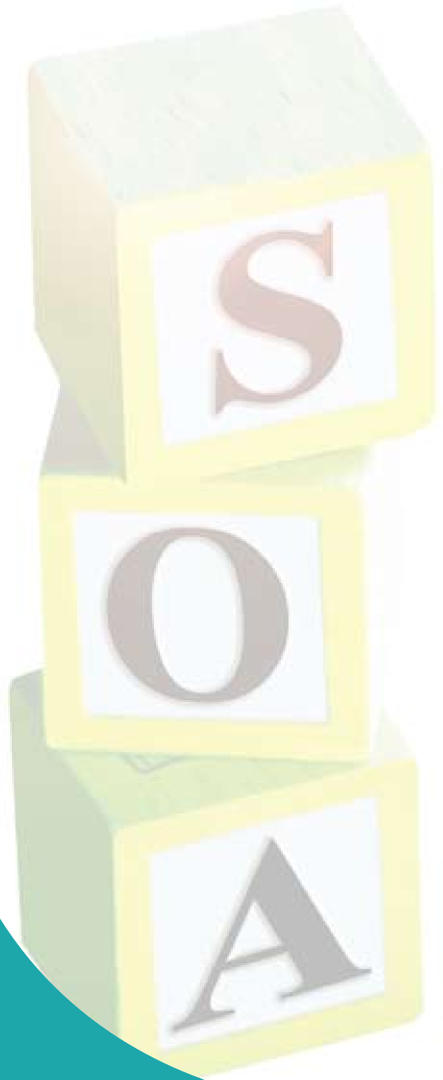


Figure 5-3. How Web services standards interact

To guard against interoperability issues and the potential slowdown in Web services adoption that might result, the vendor community has formed the independent Web Services Interoperability Organization (WS-I). WS-I's charter is to promote the interoperability of Web services implementations from different vendors. WS-I creates profiles, which are implementation guidelines for interoperability, and testing tools to verify conformance with a profile and WS-I guidelines. Currently available profiles include the Basic Profile, Attachments Profile, and Simple SOAP Binding Profile. In addition, work is under way on a Basic Security Profile. WS-I profile conformance is an important consideration for organizations that want to ensure interoperability of their Web services across different vendor platforms.



Case Study 5-2
Company: U.S. Power & Light

Company Description

U.S. Power & Light is a pseudonym for one of the country's largest power providers. The company manages a distribution grid that spans 11 states and provides electricity to 5 million customers.

Business Challenges

Several years ago, the Federal Energy Regulatory Commission mandated that U.S. Power & Light become a member of a regional transmission organization (RTO). RTOs are independent, government-regulated organizations focused on energy transmission and distribution for regions of the country. U.S. Power & Light needed to meet the RTO's stringent technical requirements in 10 months, or it risked federal fines of $1 million a day. These requirements included providing real-time and batch information on capacity, number of units online, power produced per hour, emissions, and outages directly from the company's power plants. At the same time, U.S. Power & Light needed to improve its operational efficiency to meet Sarbanes-Oxley requirements. Redundancies in the existing system required employees to enter the same information into 15 different systems for some transactions.

How Integration Addressed the Challenges

More than 100 new interfaces were required for the types of data and forms that U.S. Power & Light would be exchanging with the RTO. The company chose webMethods as the anchor of an SOA strategy that enables it to publish data in one place to support the RTO requirement. With the webMethods platform, all subscribing systems can now instantly retrieve the information they require. The solution is based on XML schemas, which are translated from the source system into U.S. Power & Light's internal canonical forms. This enables the data to be integrated into multiple systems without point-to-point data transformations. With the new system, U.S. Power & Light sends its RTO hourly schedules of where it will be delivering power and where it will be pulling power off.

Bottom-Line Business Benefits

The SOA-based webMethods platform has enabled U.S. Power & Light to be development platform-agnostic for application interfaces. Using Web services, the company can easily connect disparate systems and reuse code when automating new processes. U.S. Power & Light met its 10-month schedule and avoided $1 million per day in potential fines. In addition, it can now meet the Sarbanes-Oxley requirements. With all the integration points in its systems connected through a common platform, U.S. Power & Light can easily monitor and document the system and processes, errors, and other key areas as required for compliance. SOA and Web services were two key ingredients for this success.

Integration's Role in SOA

Although SOA is commonly associated with the creation of new systems, SOA and integration are also closely linked. A key driver of SOA is interoperability, which of course is at the core of integration. In driving a greater degree of system-to-system interoperability through the adoption of Web services standards, SOA will ease the challenge of integration. For reasons explained below, however, it will not replace the need for higher-level integration capabilities. Still, organizations can increase flexibility and reuse by taking an SOA approach to implementing integration projects. In essence, SOA makes integration easier, and integration technologies enable SOA to achieve greater productivity and utility.

So how does integration serve an essential role in SOA?

For one thing, as stated earlier, Web services on their own do not provide the full spectrum of functionality required in an enterprise SOA. Beyond the interoperability enabled by Web services, additional capabilities are required to assemble, orchestrate, and coordinate Web services into more solutions in real-world business scenarios.

For example, it will be a very long time, if ever, before all enterprise data is accessible in XML format, the data format for Web services. Different applications—built with different business requirements for different purposes—define business entities (such as customers, products, or financial securities) their own way. Consequently, translation and transformation of data into the native formats of each application is an essential component of integration today, and will remain an essential component of SOA. Different data representations—in terms of syntax (the format of the data) and semantics (the meaning of the data)—will exist as long as more than one system is involved. Since so much data is represented in non-XML formats, XML-based translation and transformation technologies such as XSLT do not suffice. Therefore, integration technologies—with their robust support for complex legacy data formats—are needed to provide SOA with a wider range of translation and transformation capabilities.

Additionally, forward-migration mechanisms are needed to bring the legacy environment—that is, the non-XML, non-Web-services-based systems that will continue to

constitute the majority of enterprise systems for a long time—into the SOA fold, since starting with a clean slate is not a viable option. This forward migration is often expressed in terms of putting Web services wrappers around legacy interfaces to make them available to the SOA world.

Though jumping from an existing legacy interface to one based on Web services may be technically feasible, it may not be desirable from a financial or risk-management perspective. Creating Web services interfaces for existing systems requires considerable development and investment. If the service will be reused often, the investment will pay off over time. For some implementations, however, getting a new solution up and running quickly is the prime requirement. When connecting to enterprise resource planning, packaged, and legacy applications, it may be faster and easier to use existing application adapters provided by an integration vendor than to recreate access via Web services interfaces. Using application adapters for tactical solutions while taking an evolutionary approach to creating Web services from existing applications will allow an organization to realize an early return on investment (ROI) from SOA implementations. Some integration vendors enable a more seamless transition by making operations exposed through an adapter automatically available as Web services.

Finally, Web services do not provide the requisite infrastructure services for organizations focused on using SOA to support a strategic initiative to optimize business processes. Process optimization requires managing the end-to-end business process, which crosses application and organizational boundaries. This is what business process management (BPM) systems do. Web services orchestration defines the flow of control for a composite application, but BPM goes further with real-time monitoring management, and process optimization through process automation, human workflow management, and process simulation.

The relationship between SOA and integration can be summed up as follows:

- SOA and integration are highly complementary. On the one hand, SOA simplifies integration by driving interoperability between systems via Web services, and by componentizing application functionality into business services, making systems more accessible to integration. On the other hand, integration technologies and solutions provide a migration path to SOA from the existing installed base of pre-SOA systems.

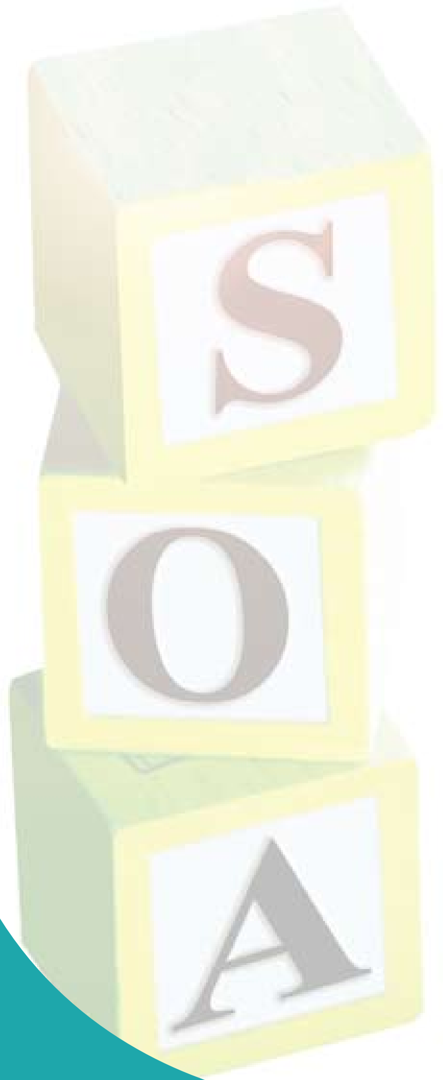
- Interoperability and integration are not equivalent. Web services promote cross-platform interoperability, but true business integration demands higher-level capabilities such as process orchestration, process monitoring, data reconciliation, and exception management. These capabilities are necessary elements of SOA-based business solutions, but they are not features provided by Web services. Integration technologies and solutions exist to fill these needs.

- The disciplines behind integration and SOA-based development are similar, and best practices developed for integration can be applied to SOA. Organizations that take a strategic approach to integration understand that success lies in appropriate methodologies, governance on the use of organizational IT assets, and dedicated organizational structures and roles. These same principles apply in practice to SOA. For example, the expertise and processes developed to manage integration interfaces—to ensure that they are well specified, meet required service levels, and so on—can be applied directly to a registry of business services. In short, the expertise and competencies developed for integration give organizations a head start with SOA.

Enterprise-Class SOA

As stated earlier, simply making Web services available on the network does not create an SOA. For an SOA to be effective in the real world, numerous infrastructure capabilities are needed—such as data translation and transformation, process orchestration, and the other integration capabilities described in the previous chapter—as well as capabilities for security, versioning, auditing, management, monitoring, and ensuring predictable quality of service. The latter capabilities are especially important for enabling SOA to address mission-critical business requirements.

It is useful therefore to think of an enterprise-class SOA as one that not only incorporates the business services underlying the applications delivered via the SOA, but also provides important infrastructure capabilities for delivering the reliability, scalability, and manageability that global organizations expect. A coherent SOA infrastructure will also help manage the complexity of the environment, without which sustainable reusability would not be possible.



Case Study 5-3
Company: State Bank

Company Description

State Bank is a pseudonym for a European bank based in Germany. With a focus on midsize enterprises, the bank offers its domestic and international customers the financing services of an internationally oriented credit institution and the investments and services of an investment bank. State Bank has more than 1,700 employees and over €70 billion in revenues.

Business Challenges

State Bank was introducing SAP's Bank Analyzer, which includes a financial database (FDB). The FDB was to be the bank's new core database, and all enterprise applications needed to be connected to it. This requirement made State Bank's previous interface model obsolete because it was time-consuming and led to a single point of failure.

How Integration Addressed the Challenges

State Bank decided that adopting an SOA would shorten the development time of the interfaces, making them reusable and easier to maintain while eliminating the single point of failure. The bank implemented webMethods Fabric as the SOA-based integration layer for connecting its legacy systems and databases to the FDB, and used XML to map all the data entering and leaving the integration layer. As a result, all of the bank's systems can now communicate with the FDB for the SAP solution.

Bottom-Line Business Benefits

The SOA-based integration solution has replaced State Bank's obsolete interface architecture with a flexible, reusable system. After only a three-month implementation phase, the new platform is already processing 170,000 transactions each month. As a result of its approach, State Bank can more easily synchronize customer information with the customer master data, obtain foreign exchange prices in real-time, and update new securities holdings after every transaction.

SOA Infrastructure

Deploying an SOA infrastructure as part of the overall SOA implementation strategy is vital if organizations are to realize the benefits promised by SOA. Although a tactical approach might be possible in the early stages of an SOA implementation, long-term ROI requires a proactive approach to creating the SOA infrastructure.

- An enterprise-class SOA infrastructure:
- Promotes architectural consistency by ensuring that Web services are deployed in a manner consistent with a service-oriented model
- Facilitates the assembly of large-scale, dynamic systems from Web services
- Provides a central point for managing and monitoring deployed services
- Delivers the performance and robustness required for business-critical systems through features such as load balancing and automatic failover between services
- Provides mechanisms whereby services can easily locate one another, regardless of where they are deployed in the network, and promotes loose coupling of business-oriented services through location transparency and dynamic binding
- Provides metadata management and registry services to support the reusability of services
- Provides a framework for creating and deploying infrastructure services—services that function between business-level services, such as transformation and filtering
- Allows services to be provisioned dynamically as business needs fluctuate
- Provides security across services, addressing needs such as client authorization
- Offers various levels of logging—for example, Web services requests and responses, session-related information, and security events—to support trend analysis, exception handling, and problem diagnosis

An important requirement of an enterprise-class SOA infrastructure is that it exist independent of the business services and applications it supports. As the figure on the next page illustrates, different Web services containers may have different capabilities (including no native Web services capabilities at all). In addition, even when different containers have similar capabilities (for example, support for a specific WS-* standard), the actual implementations may vary or be incompatible because of version differences.

As a result, the only way to achieve the appropriate consistency across different service containers for functions such as security, transactions, service monitoring, and quality of service is to externalize the SOA infrastructure from the service providers. Conceptually, this means that instead of consumers calling Web services providers directly, calls are made through an interception layer that provides a consistent set of infrastructure capabilities across service providers. In addition to enabling consistency, this layer enables the SOA infrastructure to do things like load-balance across different services and provide Web services capabilities to systems that are not natively built on Web services.

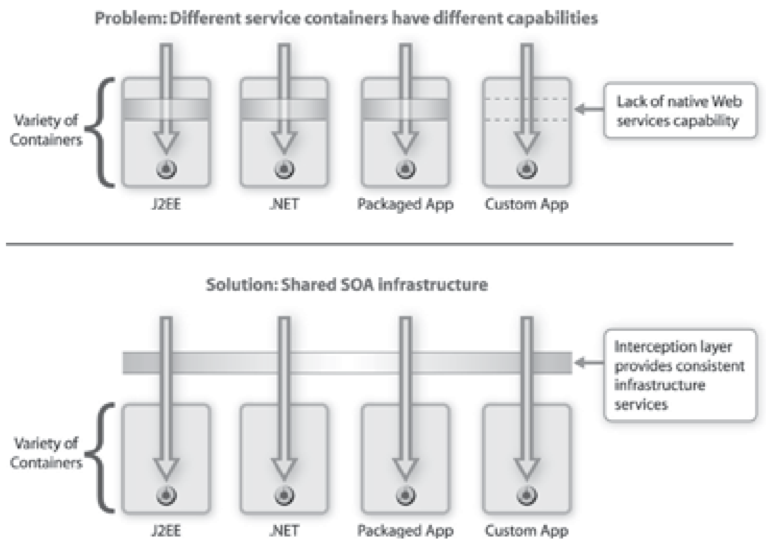
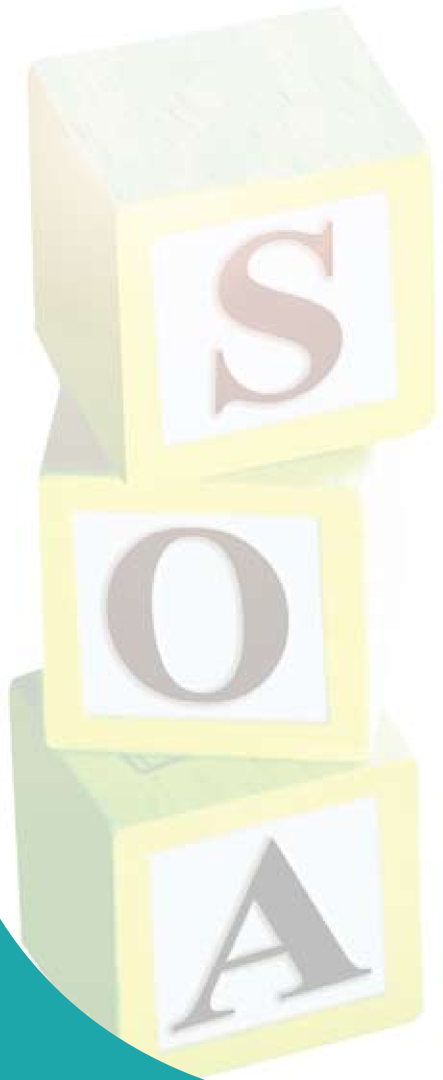

Problem: Different service containers have different capabilities
Variety of Containers
Lack of native Web services capability
J2EE
.NET
Packaged App
Custom App
Solution: Shared SOA infrastructure
Interception layer provides consistent infrastructure services
Variety of Containers
J2EE
.NET
Packaged App
Custom App

Figure 5-4. Enterprise SOA infrastructure

As explained, the SOA infrastructure should be non-intrusive to the business services themselves. It should also be capable of operating natively within supported platforms, such as an application server, and as an intermediary for nonsupported platforms, such as a packaged application or a service hosted by a third party). With this flexibility, an SOA infrastructure can bring all the services in an organization's environment under a single management scope.



SOA Governance

Providing the capabilities of an enterprise-class SOA infrastructure requires a combination of standards, technologies, policies, and procedures. Although the appropriate technologies are important, corresponding policies and procedures are just as crucial to realizing the benefits of SOA. As Web services begin to proliferate in organizations and services are reused in different applications and different areas of the organization, governance becomes an essential part of the SOA strategy. So does the need to enforce policies and procedures throughout the development and maintenance life cycle of a Web service.

One important area of SOA governance is managing reuse. As experience with object-oriented programming has shown, if ROI is to be realized, the discipline of reuse must be designed into the process. Reuse does not happen by accident. When the proliferation of Web services is unmanaged, it leads to duplication, overlap, and an overall loss of control that erodes the reusability benefits of SOA. Managing reuse means architecting services so they can be reused, monitoring how services are reused, and providing appropriate incentives for reuse. Although these issues are primarily ones of methodology and process, some vendors provide capabilities to help manage reuse. For example, by profiling the performance characteristics of a given Web service (such as average response time), a developer can make an informed decision about whether to use the service for a specific business requirement.

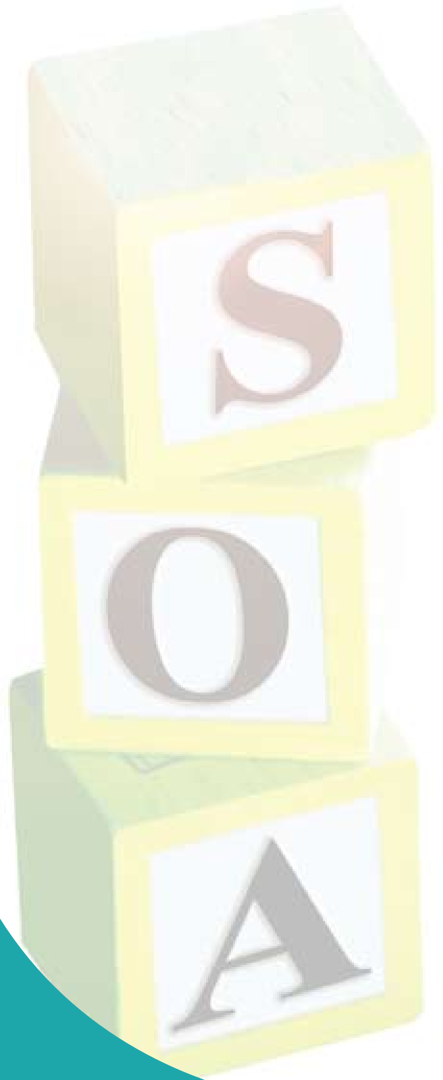
An additional governance issue relates to policy definition, implementation, and enforcement. Consider the FindCustomer service. A company may need to implement a new business policy that requires customer credit card information to be transmitted in an encrypted format to protect customer confidentiality. The developer of the service can easily ensure that the service receives and delivers only encrypted information. The challenge is communicating and enforcing the policy change with individuals who are using the earlier, unencrypted version of that service.

A centralized registry helps support the definition, implementation, and enforcement of service policies. Putting policy rules in a centralized governance repository makes them much easier to enforce across applications and platforms. For example, development policies can be enforced when a service is registered, and run-time policies can be enforced when a service is invoked. Because of this, the UDDI registry is increasingly viewed as more than just a place to catalog services; it is also a strategic asset in the SOA infrastructure from a governance perspective.

Enforcing security policies across services is another important governance issue. A Web service may be created for use within a single department but then adopted by other users, including users outside the organization. Security must be maintained across users. An ad hoc approach to Web services management provides no mechanism for implementing a security policy and enforcing compliance across the organization.

A final issue relates to the operational monitoring and management of Web services from a service-level perspective. A Web service may be used by applications across the enterprise that have different quality-of-service expectations regarding availability, performance, and security. It may be possible to monitor the performance of the service in isolation, but the more important requirement is whether the service satisfies the service-level expectations of a particular user.

When issues such as these are not addressed at an enterprise level, and tactical Web-services-based solutions begin to proliferate without effective governance, the benefits of SOA rapidly decline. As such, the more progressive software vendors offer not only technology solutions but also a broader portfolio of capabilities that includes best practices and methodologies for helping organizations realize their SOA strategies effectively.



Best Practices in SOA

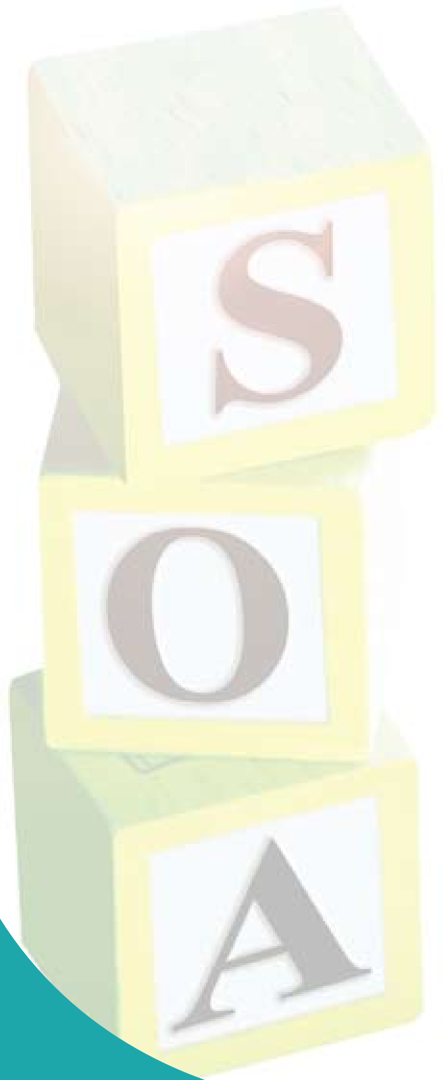
By now it should be clear that SOA is an enterprise strategy, not a technology or a tactical solution. To reap the rewards of SOA, organizations must implement an enterprise-level approach. Best practices include the following:

Create an enterprise SOA group. This group is responsible for defining the enterprise SOA architecture and roadmap, and managing the SOA's incremental implementation by working with project groups across the organization. The enterprise SOA group is similar in function and purpose to the Integration Competency Center (ICC), as described in Chapter 6, and may evolve as an outgrowth of the ICC or the Enterprise Architecture Group, if one exists.

Manage reuse. As mentioned earlier, reuse must be managed if any benefits are to be realized. Among other things, managing reuse requires planning and designing for reuse, maintaining appropriate documentation and metadata about reusable assets, having a registry of the services available for reuse, and providing active administration. Policies and incentives for reuse are also important.

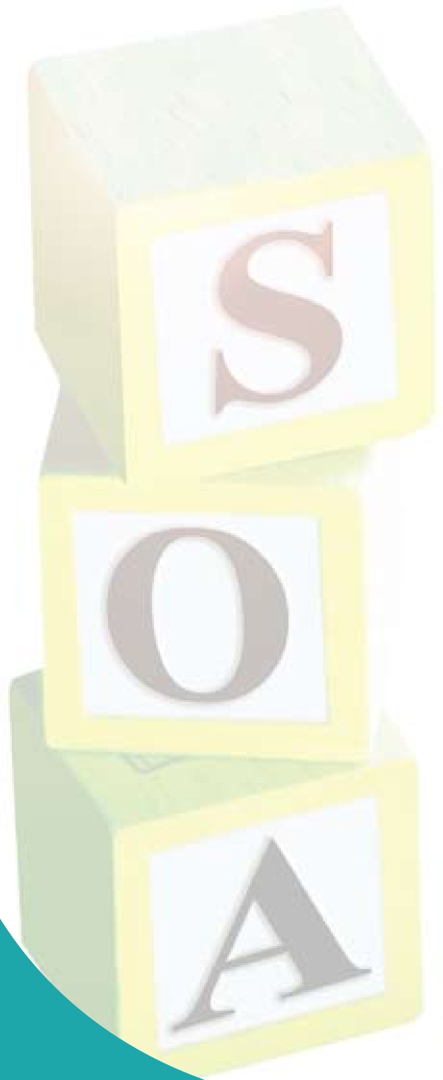
Control redundancy. Duplication of technical and business services drastically increases maintenance costs and complexity. Redundancy usually results from poor governance and a temptation to take shortcuts. These can be avoided with strong enterprise controls and best practices.

Develop SOA competency within the organization. Loosely coupled Service Oriented Architectures provide maximum business agility. However, it is not enough to provide programmers with an integrated development environment for creating Web services. Programmers need to understand how to design and build systems that follow and take advantage of the new paradigm. For example, developers should understand how to design services at the right level of granularity to maximize reuse. For long-term success, organizations need to invest in developing a core competency in SOA design and implementation.



Conclusion

SOA is the undisputed path for creating an agile IT infrastructure. Although there are many important technology considerations—including the implementation of the integration technologies represented in the ebizQ Integration Roadmap (Chapter 4)—SOA is about more than technology. The broad, successful adoption of SOA requires a proactive, managed approach that has implications for an organization's development practices, its organizational structures, and its approach to investment in applications and software infrastructure. Taking the proper strategic approach requires an initial upfront investment, but the benefits accrue as the number of deployments grows. Over the long run, a strategic approach offers a much higher ROI than what might be experienced with quick and easy tactical solutions.



Beth Gold-Bernstein

VP, ebizQ

Beth Gold-Bernstein is a recognized expert in integration technologies and SOA. She has over 20 years experience working with financial institutions, retail chains and manufacturers on planning, designing and implementing large scale distributed systems and enterprise architectures. She has developed training programs for business analysts, architects, and implementers, and speaks nationally and internationally at conferences and seminars. Beth is the author of "Enterprise Integration: the Essential Guide to Integrated Solutions," published in July 2004 by Addison Wesley. Her first book, "Designing Enterprise Client/Server Systems," was published by Prentice Hall in August 1997. Beth holds an M.S. in Computer Information Systems from Bentley College.

Gary So

VP, Office of the CTO, webMethods, Inc.

Gary So is Vice President, Office of the Chief Technology Officer, at webMethods, Inc, where he is responsible for advancing the company's status as a recognized industry thought leader. Gary has over 10 years experience in the integration field, serving previously as a system architect in corporate IT and as a director of professional services at Active Software, Inc., before joining webMethods in 2000 and spear-heading the development of the company's integration methodology. Formerly VP of Strategic Marketing, he was in charge of driving webMethods' agenda through media and analyst relations. Gary has a Masters degree in Computer Engineering from the University of Toronto.