

# The SOA Magazine

## Feature Article



### SOA Realization through Service Virtualization

by Chris Madrid

Published: September 3, 2007 (SOA Magazine Issue X: September 2007, Copyright © 2007)

[Download this article as a PDF document.](#)

*Abstract: The concept of virtualization is being widely embraced throughout enterprises and their collective data centers. From virtualized mobile devices, operating systems, and appliances to virtualized IT resources such as storage frequently realized through SAN deployments, the benefits are undeniable: resources are more highly utilized, flexibility gains result in quicker times to market; and an opportunity emerges for increased visibility by leveraging hooks in the abstraction layer. By applying virtualization to services, organizations can start to realize the potential that SOA has offered but has been challenged to deliver. This article will discuss a proven approach to service virtualization and will demonstrate it via a case study wherein a company faces common real world challenges as they adopt Web services and service-orientation.*

### Introduction

SOA strategists, enterprise architects, IT directors, and other technology decision makers have been inundated with literature and media reports proclaiming the benefits of adopting SOA. While some benefits like agility and reuse are not difficult to envision, exactly how those benefits are actually realized is less clear. Maturity models are useful in assessing how well an organization has adopted SOA but they do not provide an actionable roadmap. Consequently, a large number of enterprises find it difficult to know how to commence.

These days, a fundamental understanding of SOA concepts among project teams in typical IT organizations is relatively common. Many technology leaders are now aware of the fact that SOA does not equal the use of Web services and that the services layer is an important bridge between process and integration layers.

Often the relationship between processes and services is recursive, where a step in a particular process may invoke a Web service that in turn initiates yet another process. As organizations mature and start implementing processes on top of services and use services to invoke processes, it becomes necessary to take charge of the service lifecycle. Actively managing service identification, development, provisioning, and consumption is crucial or else organizations will find themselves maintaining the same spaghetti-style integration they are stuck with today (only this time around the spaghetti is topped off with the latest Web services standards and tools).

### Common Issues with Services

IT enterprises face many, varied challenges during the lifecycle of a typical SOA adoption. Some obstacles are purely organizational (like funding shared service development and maintenance), while others are mostly technical in nature (like true communication fidelity between multiple Web service development stacks). To overcome some of these obstacles requires a substantial mix of organizational maturity and technical solutions. Service versioning and service representation are two challenges that fall into this last category and act as examples of where service virtualization can prove itself as a strong technical solution.

## Service Versioning

Service versioning can represent a significant obstacle to achieving business agility that is typically manifested in a two-fold manner. First, the technical and management challenge of supporting provisioned consumers that are unwilling or unable to upgrade their proxy while continuing to develop innovative business functionality. These will take advantage of new technology enhancements resulting in the Web service development and maintenance teams moving through slower development cycles, thus preventing any potential gains in agility from the reuse of those services. Consequently, as line of business owners experience a decline in their own agility they will no longer willingly share their services with other lines of business. Instead of service reuse slowing because it is shared, service sharing is eliminated because of the slowing reuse.

Services today are routinely created as part of a larger custom development project for a specific line of business. Business sponsors typically allocate funds for the construction of the presentation layer, business layer, and data layers with the expectation of a certain return on investment. In order to reduce costs, subsequent lines of business application sponsors frequently expect to reuse existing services (Figure 1).

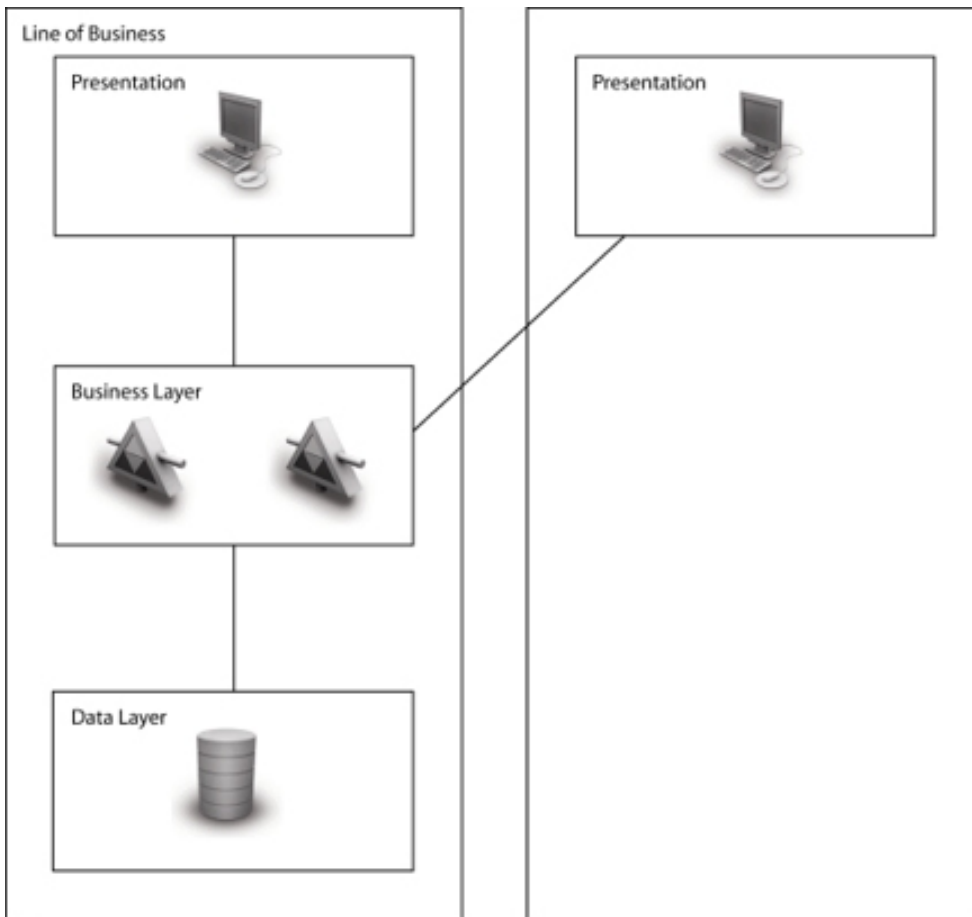


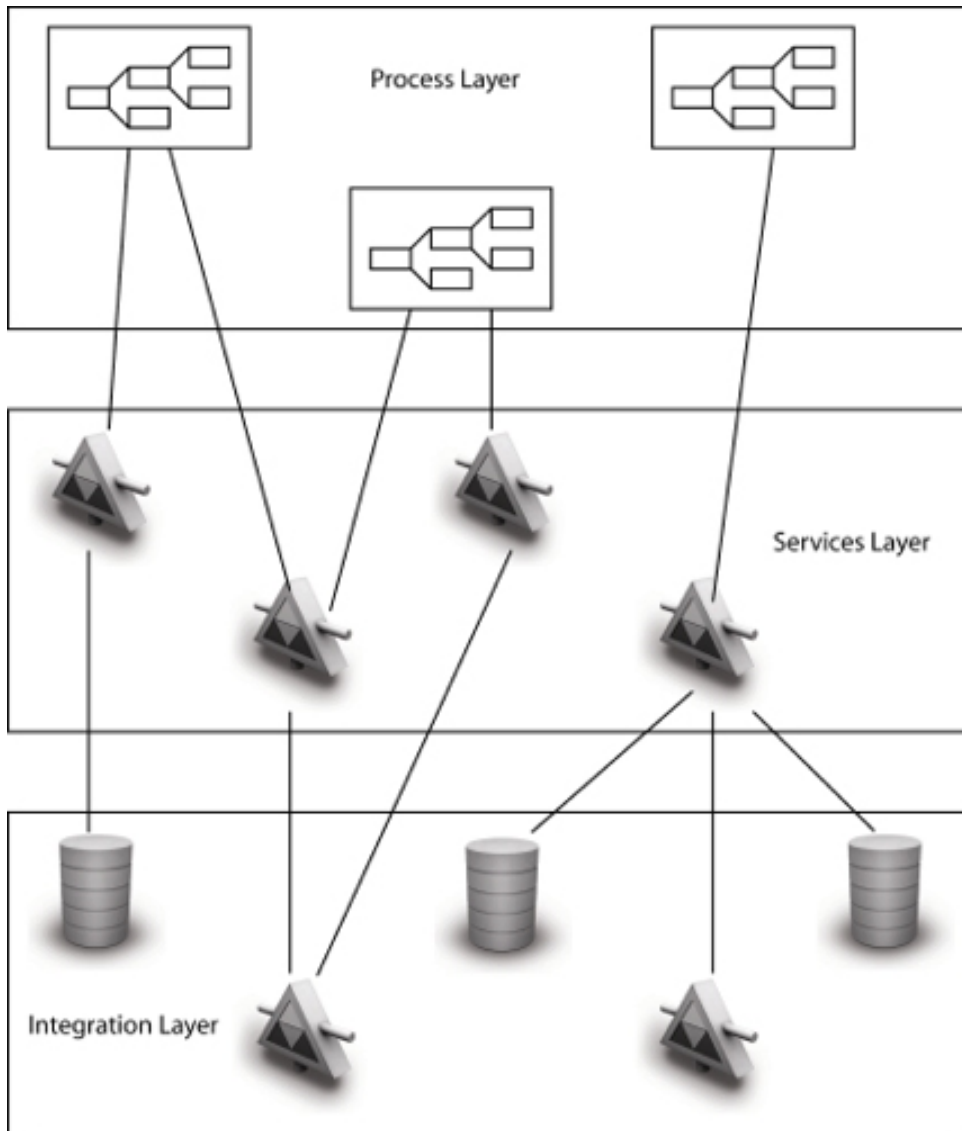
Figure 1: An *n*-tier architecture with services.

This is in line with common modern enterprise architecture principles seeking adoption of SOA and utilization of existing IT assets. As more line of business applications take advantage of existing services, the original service development team assumes a significant maintenance burden that is usually unaccounted for, thus revealing the obstacle. Eventually, line of business owners prevent additional services from being exposed to outside groups in hopes of maintaining their own agility.

## Service Representation

Service representation is more than a service description or a service definition. Its essence is the consumer's experience. Is the service too coarse or too granular? Is the required operation logically located on the appropriate endpoint? The only fundamental information that is required to invoke business logic is the address, binding, and contract of a specific operation.

At runtime everything revolves around the operation, which accepts certain messages and supports a limited set of message exchange patterns. The service is essentially an arbitrary container of operations to assist in service taxonomy construction. This attribute is favorable in circumstances where a roadmap of services can be defined in the enterprise before they are built. To obtain maximum benefit from an SOA, it cannot live in a vacuum independent of enterprise application integration (EAI) and business process management (BPM). Service analysts and architects must identify and design services that weave all three platforms into a cohesive enterprise stack.



*Figure 2: The grouping of architectural elements into logical layers.*

Accomplishing this task from a pure top-down or bottom-up approach will result in services that are too skewed to one end of the spectrum to be of practical use. In the case of a top-down strategy, where the focus is on business processes, identified services represent the ideal service and often fail to account for IT planning and available hardware, personnel, and integration methods.

With the bottom-up approach, where the focus is on service enablement, services typically mirror the existing integration interfaces. They are able to achieve some benefit from a more interoperable interface but this hasn't been the barrier to enterprise agility. Products and technologies have been available for quite some time to bridge this gap.

The true barrier is that new interfaces that align more to the business haven't been designed and implemented because of major versioning challenges. The recommended path is one that forces business-IT alignment in jointly identifying services, contracts, and an implementation roadmap.

In adhering to this joint path, service architects are free to identify three categories of services:

- Task Services

- Entity Services
- Utility Services

Task-based services allow parent processes to be implemented as services. Entity-based services represent functionality associated with business entities (customers, accounts) and can often be found aggregating data from multiple utility services that service-enable many enterprise resources, such as legacy systems and data stores.

Unfortunately, many organizations began deploying Web services well before SOA was coined and prior to the availability of structured principles for designing services [REF-1]. Some organizations have hesitated rewriting their services to follow these principles in the hopes that they will eventually realize some of the strategic SOA benefits with what they have. This, however, rarely happens. Instead, anti-patterns become more ingrained into the enterprise making it more difficult, over time, to get the services back on track.

Transitioning from abstract concepts to a concrete implementation is usually where teams looking to adopt and implement SOA are challenged the most. Service virtualization is a proven approach for helping teams manage the service lifecycle and achieve tangible and measurable SOA benefits. Let's take a closer look at what service virtualization is all about.

### Service Virtualization as a Solution

At its most basic form, virtualization is an abstraction layer that decouples a resource consumer from the provider of those resources. Virtualized resources allow a single provider to support multiple consumers to achieve higher agility and reuse of those resources. In the virtual machine scenario, the physical hardware is abstracted from the operating system creating unique opportunities for design optimization.

Capable of supporting multiple running instances of different operating systems on the same machine, virtual machines deliver unparalleled utilization of extra processing cycles in the data center and developer workstations. Service virtualization realizes many of the professed SOA advantages by abstracting the true service address, binding, and contract necessary to invoke business functionality at a given endpoint. An abstraction layer is also a potential platform for additional behaviors.

*Address virtualization* abstracts the URI where the endpoint actually sits from service consumers and allows messages to be delivered per metadata to different endpoints. Coupled with content-based routing behavior, for example, address virtualization could be used to manage a service level agreement (SLA). For instance, a Web farm can be dedicated to silver customers and another more powerful and robust Web farm can be made available for gold customers. Consumers would appear to invoke the same endpoint, but in reality messages would be routed to the appropriate endpoint, thus ensuring an appropriate level of service that corresponds to the importance of the customer.

*Binding virtualization* abstracts the actual communication protocols supported by the endpoint to more easily allow providers to adopt newer Web service development stacks while still supporting existing service consumers. Contract virtualization abstracts WSDL and XSD definitions so services can be discovered. For example, a service may be dynamically comprised of operations from disparate services to form a single service that may serve as a better representation of a task or entity service. Such seemingly straightforward capabilities are extremely powerful and organizations may be willing to undergo major infrastructure upgrades to better support a virtualized service inventory.

Many vendors profess service virtualization as a capability of their intermediary, agents, or bus. Usually they limit the concept to contract virtualization and attribute address virtualization and binding virtualization as a feature of their product. It is important to understand that the concept exists independently from any specific implementation architecture. Certain architecture designs are supported by a stronger implementation and serve as a base for security, monitoring, and caching of value-added services. Service virtualization implemented over the service intermediary pattern ideally achieves these goals in a non-invasive manner and can be easily applied to existing production deployments. Moreover, the centralized nature of service intermediaries makes them an excellent platform for service repository and service registry capabilities.

To gain a real world perspective of this approach, let's explore the case study example.

### Sunset Communications

As an established telecom company, Sunset Communications offers a range of telephone and high-speed Internet services and products. Call center agents have access to a smart client application that consumes an Account entity Web service that aggregates telephone billing data from a legacy iSeries application and high-speed Internet billing data from an Oracle database. As shown in Figure 4, customers may also access their billing information on the Internet.

The Web servers hosting the Web application consume the same Account service, which complies with the WS-I Basic Profile 1.1 and thus only accepts a SOAP 1.1 envelope over the HTTP/1.1 protocol and is described via a WSDL 1.1 definition. (This represents version v1.0 of the Account service.)

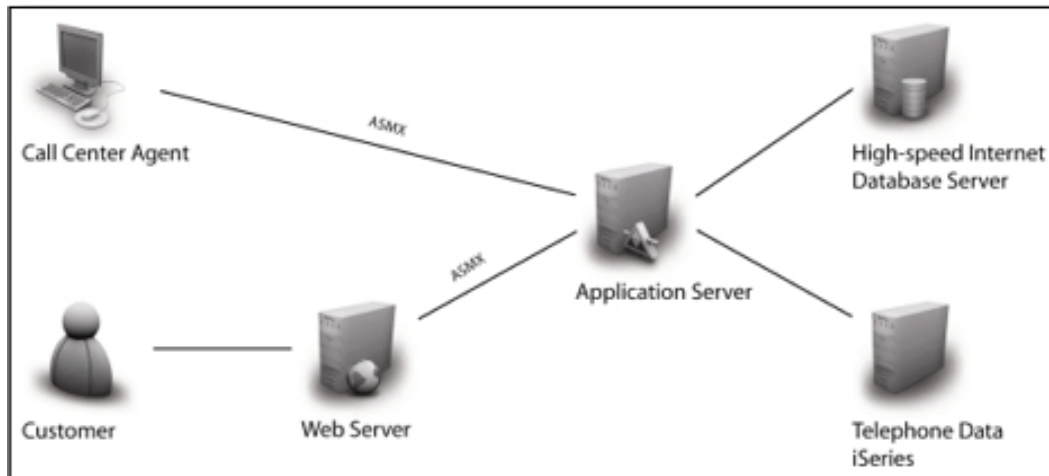


Figure 3: The existing Sunset Communications architecture.

Competitive forces in the industry are pushing the business to expand and begin offering TV and VOIP solutions over their high-speed communications infrastructure. To deliver these additional products, the company must be able to bill customers and respond to customer billing inquiries. This functionality will require the development of a new service or, more appropriately, the revision of the existing Account service to include TV and VOIP billing information. The marketing department also wants the ability to bundle offers and have these bundles represented on customer billing statements.

The Web development team is allocated to creating new product information pages and does not have the necessary resources to devote to consuming the modified Account service. Rather than delay the product launch, the business decides that solely supporting TV and VOIP billing inquiries in the call center was acceptable for the first few months in order to maintain a market leading position. To support this scenario, the Account service team must continue to support their v1.0 consumers and now deploy a v1.1 Account service that includes TV and VOIP billing information in its response. Furthermore, to comply with a recently released policy, new services must be built using WS-Security and SOAP 1.2.

Virtualization of the Account service through a service intermediary would allow the service development team to upgrade its codebase to Windows Communication Foundation (WCF) to meet the requirements of WS-Security and SOAP 1.2 for new consumers while introducing a modified contract that would include TV and VOIP billing information. Leveraging protocol conversions and message transformations, the intermediary is able to make the revised service appear unmodified to existing consumers.

Coupled with metadata configuration information, the intermediary (also acting as a registry) would no longer allow the v1.0 service to be discovered or provisioned even though it may continue to be invoked by consumers previously provisioned. Version v1.1 of the service is discoverable and also able to be provisioned and consumed. Only a single version of a service should be discoverable even though several versions of that service may remain consumable. Because this information is stored in metadata, it creates opportunities for service owners to actively manage consumers of older versions, urging them to upgrade and, at a minimum, identifying a “sunset” strategy and roadmap.

To support the marketing department’s requirements, IT needs to provide a service capable of managing the list of accounts that already subscribe to Sunset’s telephone and high-speed Internet offerings. The Account service already exposes an out-of-band operation to return a list of accounts for use during the bill printing process.

Service architects looking at existing and new requirements feel the organization could be better served by the

creation of a Billing service that would act as a façade to the billing operations supported by the Account service. Consumers of the Billing service would have a clear indication of the intended use of the operations and the operations available would represent a substantial subset of the operations supported by the Account service. Manually implementing a Billing façade service is straightforward, but would introduce another touch point for maintenance. A more robust approach would be to utilize the contract abstraction capabilities of service virtualization.

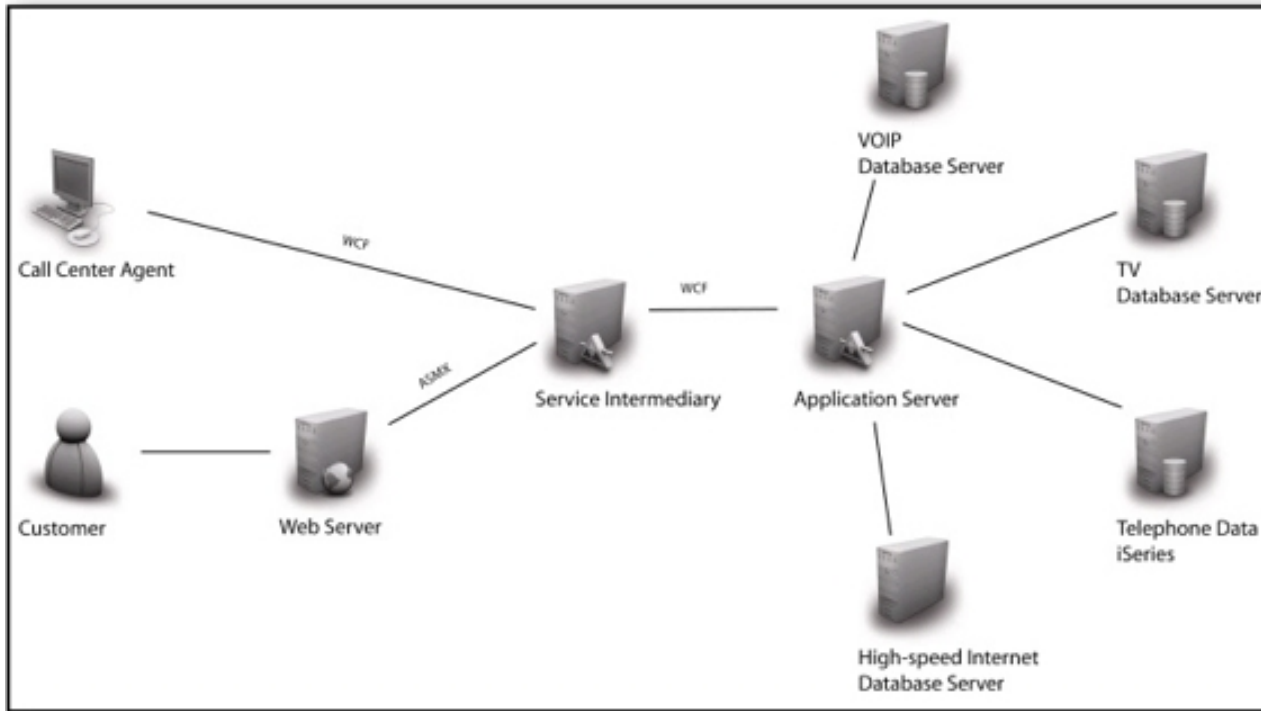


Figure 4: The proposed Sunset architecture.

At its heart, service virtualization is driven by metadata. Managed services are required to make themselves known to the service intermediary. This rich metadata can be used by the service intermediary to construct a virtual WSDL definition with operations culled from managed services. Sunset is essentially able to craft their Billing task service through configuration.

Effectively using service metadata to drive how virtualized services are provisioned and consumed enables our fictional IT team to accomplish the following:

- The discoverability of the v1.0 service can be discontinued.
- New consumers may only discover and be provisioned on the v1.1 service.
- Modifications to provisioned v1.0 consumers are unnecessary with protocol and message transformations.
- Developers are free to upgrade the Web services stack to support new IT requirements.
- Construction of a Billing task service is accomplished entirely through configuration.

## Conclusion

It is easily conceived how organizations facing these challenges would be able to achieve concrete results from abstract concepts. Unfortunately, many companies are stuck at a point where senior leadership has mandated SOA adoption and IT management is left struggling with how to apply abstract SOA concepts within the data center and to its hardware, software, and personnel resources.

The ideas presented in this article have been implemented in numerous real world SOA deployments with great success. However in much the same manner that SOA benefits should be expected to come well into the adoption (or subsequent to adoption), a service virtualization implementation presents its own challenges with the rewards to follow. Once IT is comfortable with the concepts and capabilities of implementing managed services, the organization will have a strong foundation for achieving and maintaining business-IT alignment, even as the business strives to

innovate and stay ahead of its competitors.

## References

[REF-1] "SOA: Principles of Service Design", Thomas Erl, Prentice Hall/PearsonPTR, 2007.

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[Home](#) [Past Issues](#) [Contributors](#) [What is SOA?](#) [SOA Books](#) [Legal](#) [RSS](#)

Copyright © 2006-2007 SOA Systems Inc.

All Rights Reserved