nderstanding Service-Oriented rchitecture

/2009 • 22 minutes to read

s article

l Sprott and Lawrence Wilkes
Forum

ıry 2004

nary: Gives a concise explanation of service-oriented architecture, what it is, and how ects what architects, CIOs, project managers, business analysts, and lead developers 3 printed pages)

ents

nary

roduction

ms probable that eventually most software capabilities will be delivered and
ımed as services. Of course they may be implemented as tightly coupled systems, but
oint of usage—to the portal, to the device, to another endpoint, and so on, will use a
:e-based interface. We have seen the comment that architects and designers need to
utious to avoid everything becoming a service. We think this is incorrect and muddled
ing. It might be valid right now given the maturity of Web Service protocols and
ology to question whether everything is implemented using Web services, but that
n't detract from the need to design everything from a service perspective. The service
major construct for publishing and should be used at the point of each significant
ace. service-oriented architecture allows us to manage the usage (delivery,
sition, consumption, and so on) in terms of, and in sets of, related services. This will
big implications for how we manage the software life cycle—right from specification
quirements as services, design of services, acquisition and outsourcing as services,
management of services, and so on.

time, the level of abstraction at which functionality is specified, published and or
ımed has gradually become higher and higher. We have progressed from modules, to
ts, to components, and now to services. However in many respects the naming of
s unfortunate. Whilst SOA is of course about architecture, it is impossible to constrain
iscussion to architecture, because matters such as business design and the delivery
:ss are also important considerations. A more useful nomenclature might be Service
tation (or SO). There are actually a number of parallels with object orientation (or OO)
omponent-based development (CBD):

Like objects and components, services represent natural building blocks that allow us to organize capabilities in ways that are familiar to us.
Similarly to objects and components, a service is a fundamental building block that

1. Combines information and behaviour.
2. Hides the internal workings from outside intrusion.
3. Presents a relatively simple interface to the rest of the organism.

Where objects use abstract data types and data abstraction, services can provide a similar level of adaptability through aspect or context orientation.

Where objects and components can be organized in class or service hierarchies with

inherited behaviour, services can be published and consumed singly or as hierarchies and or collaborations.

any organizations, the logical starting place for investigating service-oriented ecture is the consideration of Web services. However Web services are not inherently e oriented. A Web service merely exposes a capability that conforms to Web services cols. In this article we will identify the characteristics of a well formed service, and de guidance for architects and designers on how to deliver service oriented cations.

nciples and Definitions

ng around we see the term or acronym SOA becoming widely used, but there's not a precision in the way that it's used. The World Wide Web Consortium (W3C) for ple refers to SOA as '*A set of components which can be invoked, and whose interface iptions can be published and discovered*'. We see similar definitions being used here; it's a very technical perspective in which architecture is considered a technical mentation. This is odd, because the term architecture is more generally used to ibe a style or set of practices—for example the style in which something is designed onstructed, for example Georgian buildings, Art Nouveau decoration or a garden by lwin Lutyens and Gertrude Jekyll .

believes a wider definition of service-oriented architecture is required. In order to this definition, let's start with some existing definitions, and compare some W3C ngs with CBDI recommendations. We'll begin by looking at definitions of basic Service epts.

ice

A Component capable of performing a task. A WSDL service: A collection of end points (W3C).
A type of capability described using WSDL (CBDI).

rvice Definition

A vehicle by which a consumer's need or want is satisfied according to a negotiated contract (implied or explicit) which includes Service Agreement, Function Offered and so on (CBDI).

rvice Fulfillment

An instance of a capability execution (CBDI).

service

A software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a format that machines can process (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with XML serialization in conjunction with other Web-related standards (W3C).
A programmatic interface to a capability that is in conformance with WSnn protocols (CBDI).

these definitions, it will be clear that the W3C have adopted a somewhat narrower oach to defining services and other related artefacts than CBDI. CBDI differs slightly ar as not all Services are Components, nor do they all perform a task. Also CBDI nmends it is useful to manage the type, definition and fulfilment as separate items. ever it is in the definition of SOA that CBDI really parts company with the W3C.

ice-Oriented Architecture:

A set of components which can be invoked, and whose interface descriptions can be published and discovered (W3C).

rejects this definition on two counts: First the components (or implementations) will not be a set. Second the W3C definition of architecture only considers the mented and deployed components, rather than the science, art or practice of building rchitecture. CBDI recommends SOA is more usefully defined as:

olicies, practices, frameworks that enable application functionality to be provided and med as sets of services published at a granularity relevant to the service consumer. es can be invoked, published and discovered, and are abstracted away from the mentation using a single, standards-based form of interface. (CBDI)

defines SOA as a style resulting from the use of particular policies, practices and eworks that deliver services that conform to certain norms. Examples include certain ularity, independence from the implementation, and standards compliance. What definitions highlight is that any form of service can be exposed with a Web services ace. However higher order qualities such as reusability and independence from mentation, will only be achieved by employing some science in a design and building ss that is explicitly directed at incremental objectives beyond the basic

operability enabled by use of Web services.

A Basics

ould be easy to conclude that the move to Service Orientation really commenced with services—about three years ago. However, Web services were merely a step along a longer road. The notion of a service is an integral part of component thinking, and it ar that distributed architectures were early attempts to implement service-oriented ecture. What's important to recognize is that Web services are part of the wider e that is SOA. The Web service is the programmatic interface to a capability that is in rmance with WSnn protocols. So Web services provide us with certain architectural cteristics and benefits—specifically platform independence, loose coupling, self iption, and discovery—and they can enable a formal separation between the provider onsumer because of the formality of the interface.

e is the important concept. Web Services are the set of protocols by which Services can blished, discovered and used in a technology neutral, standard form.

t Web services are not a mandatory component of a SOA, although increasingly they ecome so. SOA is potentially much wider in its scope than simply defining service mentation, addressing the quality of the service from the perspective of the provider he consumer. You can draw a parallel with CBD and component technologies. COM UML component packaging address components from the technology perspective, but or indeed Component-Based Software Engineering (CBSE), is the discipline by which nsure you are building components that are aligned with the business. In the same Web services are purely the implementation. SOA is the approach, not just the service alent of a UML component packaging diagram.

of these SOA characteristics were illustrated in a recent CBDI report, which compared services published by two dotcom companies as alternatives to their normal browser-d access, enabling users to incorporate the functionality offered into their own cations. In one case it was immediately obvious that the Web services were ingful business services—for example enabling the Service Consumer to retrieve s, generate lists, or add an item to the shopping cart.

ntrast the other organization's services are quite different. It implemented a general ose API, which simply provides Create, Read, Update, and Delete (CRUD) access to database through Web services. While there is nothing at all wrong with this mentation, it requires that users understand the underlying model and comply with usiness rules to ensure that your data integrity is protected. The WSDL tells you

ng about the business or the entities. This is an example of Web services without SOA.

's not just an architecture of services seen from a technology perspective, but the es, practices, and frameworks by which we ensure the right services are provided and rmed.

nat we need is a framework for understanding what constitutes a good service. If, as ave seen in the previous example, we have varying levels of usefulness, we need some ples of Service Orientation that allow us to set policies, benchmarks and so on.

an discern two obvious sets here:

Interface related principles—Technology neutrality, standardization and consumability.
Design principles—These are more about achieving quality services, meeting real business needs, and making services easy to use, inherently adaptable, and easy to manage.

estingly the second set might have been addressed to some extent by organizations nave established mature component architectures. However it's certainly our rience that most organizations have found this level of discipline hard to justify. While quality components have been created perhaps for certain core applications where is a clear case for widespread sharing and reuse, more generally it has been hard to what has been perceived as an investment cost with a short term return on tment.

ever when the same principles are applied to services, there is now much greater eness of the requirements, and frankly business and IT management have undergone ep learning curve to better understand the cost and benefits of IT systems that are not ned for purpose. Here we have to be clear—not all services need all of these cteristics; however it is important that if a service is to be used by multiple consumers, typically the case when a SOA is required), the specification needs to be generalized, ervice needs to be abstracted from the implementation (as in the earlier dotcom case), and developers of consumer applications shouldn't need to know about the rlying model and rules. The specification of obligations that client applications must needs to be formally defined and precise and the service must be offered at a ant level of granularity that combines appropriate flexibility with ease of assembly into usiness process.

1 shows principles of good service design that are enabled by characteristics of either

services or SOA.

1. Web services and SOA

bled by b services	*Technology neutral*	Endpoint platform independence.
	Standardized	Standards-based protocols.
	Consumable	Enabling automated discovery and usage.
bled by SOA	*Reusable*	Use of Service, not reuse by copying of code/implementation.
	Abstracted	Service is abstracted from the implementation.
	Published	Precise, published specification functionality of service interface, not implementation.
	Formal	Formal contract between endpoints places obligations on provider and consumer.
	Relevant	Functionality presented at a granularity recognized by the user as a meaningful service.

principles summarized in Table 1 are complied with, we get some interesting benefits:

There is real synchronization between the business and IT implementation perspective. For many years, business people haven't really understood the IT architecture. With well designed services we can radically improve communications with the business, and indeed move beyond alignment and seriously consider convergence of business and IT processes.

A well formed service provides us with a unit of management that relates to business usage. Enforced separation of the service provision provides us with basis for understanding the life cycle costs of a service and how it is used in the business.

When the service is abstracted from the implementation it is possible to consider various alternative options for delivery and collaboration models. No one expects that, at any stage in the foreseeable future, core enterprise applications will be acquired purely by assembling services from multiple sources. However it is entirely realistic to assume that certain services will be acquired from external sources because it is more appropriate to acquire them. For example authentication services,

a good example of third party commodity services that can deliver a superior service

because of specialization, and the benefits of using a trusted external agency to improve authentication.

ocess Matters

dicated earlier, CBDI advises that good SOA is all about style—policy, practice and eworks. This makes process matters an essential consideration.

t some of the benefits of services might have been achieved by some organizations components, there are relatively few organizations that rigorously enforce the ation of provision and consumption throughout the process. This gets easier with es because of the formality of the interface protocols, but we need to recognize that eparation needs managing. For example it's all too easy to separate the build sses of the service and the consumer, but if the consumer is being developed by the team as the service then it's all too easy to test the services in a manner that reflects rstanding of the underlying implementation.

SOA it is critical to implement processes that ensure that there are at least two different eparate processes—for provider and consumer.

ever, current user requirements for seamless end-to-end business processes, a key for using Web Services, mean that there will often be clear separation between the ding and consumer organizations, and potentially many to many relationships where participant has different objectives but nevertheless all need to use the same service. ecommendation is that development organizations behave like this, even when both roviding and consuming processes are in-house, to ensure they are properly ning services that accommodate future needs

ne consumer, the process must be organized such that only the service interface ers, and there must be no dependence upon knowledge of the service mentation. If this can be achieved, considerable benefits of flexibility accrue because ervice designers cannot make any assumptions about consumer behaviours. They to provide formal specifications and contracts within the bounds of which consumers se the service in whatever way they see fit. Consumer developers only need to know e the service is, what it does, how they can use it. The interface is really the only thing nsequence to the consumer as this defines how the service can be interacted with.

arly, whilst the provider has a very different set of concerns, it needs to develop and er a service that can be used by the Service Consumer in a completely separate

ss. The focus of attention for the provider is therefore again the interface—the

iption and the contract.

ner way of looking at this is to think about the nature of the collaboration between der and consumer. At first sight you may think that there is a clear divide between mentation and provisioning, owned by the provider, and consumption, owned by the umer. However if we look at these top level processes from the perspective of orations, then we see a very different picture.

we have is a significant number of process areas where (depending on the nature of ervice) there is deep collaboration between provider and consumer. Potentially we a major reengineering of the software delivery process. Although we have two ary *parties* to the service-based process, we conclude there are three major *process* which we need to manage. Of course these decompose, but it seems to us that the ving are the primary top level processes.

The process of delivering the service implementation.

- 'Traditional' Development
- Programming
- Web Services automated by tools

The provisioning of the service—the life cycle of the service as a reusable artefact.

- Commercial Orientation
- Internal and External View
- Service Level Management

The consumption process.

- Business Process Driven
- Service Consumer could be internal or external
- Solution assembly from Services, not code
- Increasingly graphical, declarative development approach
- Could be undertaken by business analyst or knowledge worker

dvantage of taking this view is that the collaborative aspects of the process are arily contained in the provisioning process area. And the provisioning area is incredibly rtant because the nature of the agreement has a major influence on the process rements. There are perhaps two major patterns for designing consumer/provider orations:

Negotiated—Consumer and Provider jointly agree service When new services are developed though, there is an opportunity for both provider and consumer to agree what and how the services should work. In industries where there are many participants all dealing with each other, and where services are common to many

providers, it is essential that the industry considers standardizing those services. Examples include:

- Early adopters
- New Services
- Close partners
- Industry initiative—forming standards
- Internal use

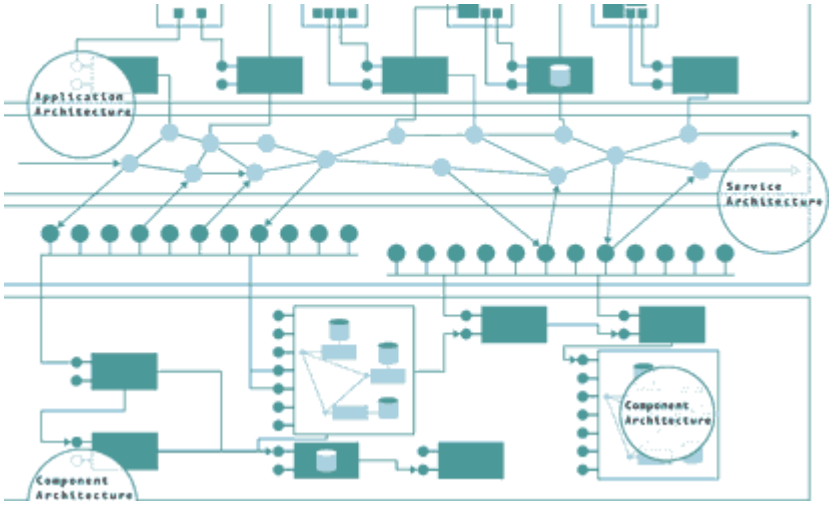
Instantiated—This is it. Take it or leave it One party in the collaborative scenario might simply dictate the services that must be used. Sometimes the service will already exist. You just choose to use it, or not. Examples include:

- Dominant partner
- Provider led—Use this service or we can't do business
- Consumer led—Provide this service or we can't do business
- Industry initiative—standards compliance
- Existing system/interface

:hitectures

›rocess view that we have examined at is a prerequisite to thinking about the type of :ecture required and the horizons of interest, responsibility and integrity. For SOA are three important architectural perspectives as shown in *Figure 1*.

The Application Architecture. This is the business facing solution which consumes services from one or more providers and integrates them into the business processes.
The Service Architecture. This provides a bridge between the implementations and the consuming applications, creating a logical view of sets of services which are available for use, invoked by a common interface and management architecture.
The Component Architecture. This describes the various environments supporting the implemented applications, the business objects and their implementations.



e 1. Three Architectural Perspectives

architectures can be viewed from either the consumer or provider perspective. Key
architecture is that the consumer of a service should not be interested in the
mentation detail of the service—just the service provided. The implementation
ecture could vary from provider to provider yet still deliver the same service. Similarly
rovider should not be interested in the application that the service is consumed in.
unforeseen applications will reuse the same set of services.

onsumer is focused on their application architecture, the services used, but not the
of the component architecture. They are interested at some level of detail in the
al business objects that are of mutual interest, for example provider and consumer
to share a view of what an order is. But the consumer does not need to know how the
component and database are implemented.

arly, the provider is focused on the component architecture, the service architecture,
ot on the application architecture Again, they both need to understand certain
nation about the basic applications, for example to be able to set any sequencing
and pre and post conditions. But the provider is not interested in every detail of the
uming application.

e Service Architecture

e core of the SOA is the need to be able to manage services as first order deliverables.
ne service that we have constantly emphasized that is the key to communication
een the provider and consumer. So we need a Service Architecture that ensures that
es don't get reduced to the status of interfaces, rather they have an identity of their
and can be managed individually and in sets.
developed the concept of the Business Service Bus (BSB) precisely to meet this need.

SB is a logical view of the available and used services for a particular business ain, such as Human Resources or Logistics. It helps us answer questions such as:

What service do I need?
What services are available to me?
What services will operate together? (common semantics, business rules)
What substitute services are available?
What are the dependencies between services and versions of services?

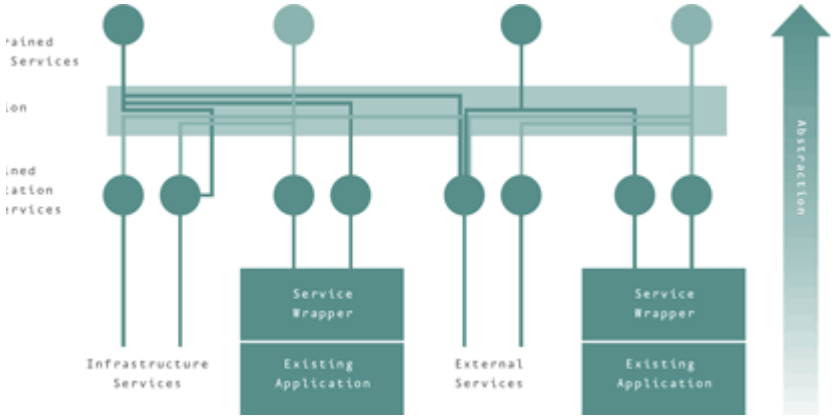
r than leaving developers to discover individual services and put them into context, the ess Service Bus is instead their starting point that guides them to a coherent set that has assembled for their domain.

urpose of the BSB is so that common specifications, policies, etc can be made at the evel, rather than for each individual service. For example, services on a bus should all v the same semantic standards, adhere to the same security policy, and all point to the global model of the domain. It also facilitates the implementation of a number of non, lower-level business infrastructure services that can be aggregated into other r level business services on the same bus (for example, they could all use the same uct code validation service). Each business domain develops a vocabulary and a ess model of both process and object.

question for the Service Architecture is 'What is the scope of the service that is shed to the Business Service Bus?' A simplistic answer is 'At a business level of action'. However this answer is open to interpretation—better to have some heuristics ensure that the service is the lowest common denominator that meets the criteria of ess, and is consumer oriented, agreed, and meaningful to the business. The key point is that there is a process of aggregation and collaboration that should probably en separately from the implementing component as illustrated in *Figure 2*. By making arate, there is a level of flexibility that allows the exposed service(s) to be adjusted ut modifying the underlying components. In principle, the level of abstraction will be oped such that services are at a level that is relevant and appropriate to the umer. The level might be one or all of the following:

Business Services
Service Consumer Oriented
Agreed by both Provider and Consumer
Combine low-level implementation-based services into something meaningful to business
Coarser Grained
Suitable for External Use

Conforms to pre-existing connection design



e 2. Levels of Abstraction

e SOA Platform

ey to separation is to define a virtual platform that is equally relevant to a number of latforms. The objective of the virtual platform is to enable the separation of services the implementation to be as complete as possible and allow components built on us implementation platforms to offer services which have no implementation ndency.

irtual SOA platform comprises a blueprint which covers the development and mentation platforms. The blueprint provides guidance on the development and mentation of applications to ensure that the published services conform to the same structural principles that are relevant to the management and consumer view of the es.

a number of different applications can all share the same structure, and where the onships between the parts of the structure are the same, then we have what might be a common architectural style. The style may be implemented in various ways; it t be a common technical environment, a set of policies, frameworks or practices. ple platform components of a virtual platform include:

Host environment
Consumer environment
Middleware
Integration and assembly environment
Development environment
Asset management
Publishing & Discovery

Service level management
Security infrastructure
Monitoring & measurement
Diagnostics & failure
Consumer/Subscriber management
Web service protocols
Identity management
Certification
Deployment & Versioning

e Enterprise SOA

ptimum implementation architecture for SOA is a component-based architecture.
will be familiar with the concepts of process and entity component, and will
rstand the inherent stability and flexibility of this component architecture, which
de a one to one mapping between business entities and component implementations.
prise SOA (ESOA) brings the two main threads—Web services and CBD (or CBSE)—
her. The result is an enterprise SOA that applies to both Web services made available
nally and also to core business component services built or specified for internal use.
eyond the scope of this article to explore ESOA in more depth. For more on this topic
is a five part CBDI Report Series on Enterprise SOA.

mmary

oal for a SOA is a world wide mesh of collaborating services, which are published and
ble for invocation on the Service Bus. Adopting SOA is essential to deliver the
ess agility and IT flexibility promised by Web Services. These benefits are delivered
y just viewing service architecture from a technology perspective and the adoption of
Service protocols, but require the creation of a Service Oriented Environment that is
d on the following key principals we have articulated in this article;

Service is the important concept. Web Services are the set of protocols by which
Services can be published, discovered and used in a technology neutral, standard
form.
SOA is not just an architecture of services seen from a technology perspective, but the
policies, practices, and frameworks by which we ensure the *right* services are provided
and consumed.

With SOA it is critical to implement processes that ensure that there are at least two

different and separate processes—for provider and consumer.
Rather than leaving developers to discover individual services and put them into context, the Business Service Bus is instead their starting point that guides them to a coherent set that has been assembled for their domain.

ırther guidance on planning and managing the transition to Web Services and SOA, are providing the 'Web Services Roadmap', a set of resources that are freely available p://roadmap.cbdiforum.com/.

ıt the Authors

l Sprott, CEO and Principal Analyst, CBDI Forum is a software industry veteran, a well-n commentator and analyst specializing in advanced application delivery. Since 1997 s founded and led CBDI, an independent analyst firm and thinktank, providing a focus e industry on best practice in business software creation, reuse and management. l can be reached at david.sprott@cbdiforum.com.

ence Wilkes, Technical Director and Principal Analyst, CBDI Forum is a frequent cer, lecturer and writer on Web services, Service-Oriented Architecture, Component-d Development and Application Integration approaches. Lawrence has over 25 years ience in IT working both for end user organizations in various industries, as well as for are vendors and as a consultant. Lawrence can be reached at nce.wilkes@cbdiforum.com.

ırticle was published in the Architecture Journal, a print and online publication uced by Microsoft. For more articles from this publication, please visit the Architecture al website.