Service Oriented Architecture

TABLE OF CONTENTS

Executive summary

Service Oriented Architecture (SOA) is currently a popular subject with no consensus or standardized reference model to define it. While many believe that Web Services[1] or ebXML[2] are SOA, they are in fact, specialized implementations of SOA.

If SOA is truly an architecture, it must be definable as one. The authors of this paper have started standards work to define an SOA reference model as part of an OASIS[3] project.

This paper examines SOA, its history, business drivers and the standards that may be used to implement it. It attempts to define SOA and reviews basic and extended models of SOA in use today. For the purposes of this paper, SOA is defined by abstracting the common concepts and elements from architectures and standards that claim to be service oriented. The localized definition of SOA is therefore subject to change in the future.

Purpose

This paper examines many implementations of a Service Oriented Architecture, identifies its most common concepts, and records the authors' opinions on these concepts.

Audience

This paper is intended for technical readers who may wish to implement a Service Oriented Architecture, analysts and the technical press community.

Note: Specifications and protocols mentioned in this paper are for reference only. The authors do not intend to advocate or pass judgment on them.

Conventions

Important terms appear in italics. Definitions are lexically (locally) scoped to this paper and may conflict with other definitions. This paper does not impose any normative constraints on the semantics of these terms.

Adobe

Introduction to Service Oriented Architecture

Service Oriented Architecture (SOA) is an evolution of the *Component Based Architecture*[4], *Interface Based Design (Object Oriented)* and *Distributed Systems* of the 1990s, such as DCOM[5], CORBA® [6], J2EE™ [7] and the Internet in general.

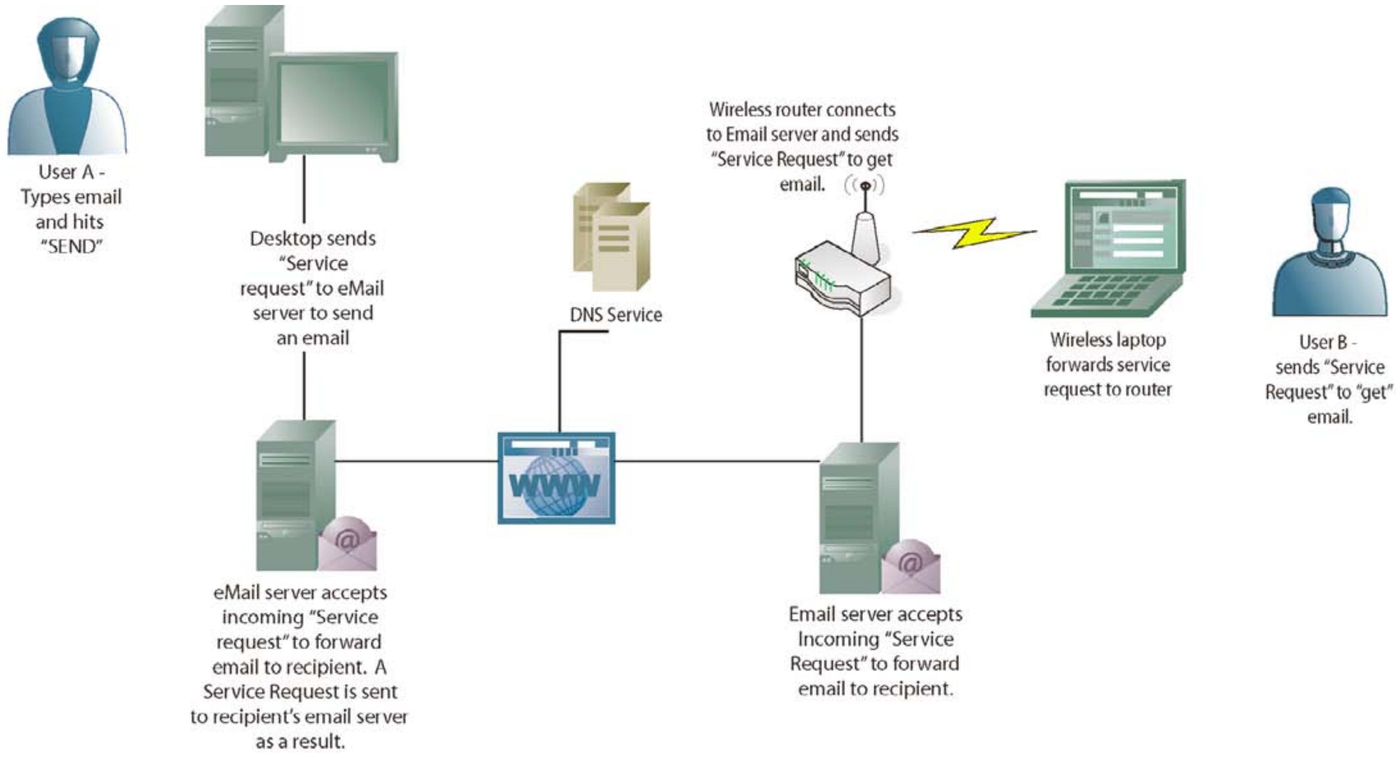
SOA does not specifically mean *Web Services, .NET ™, J2EE, CORBA* or *ebXML*. These are instead specialized SOA implementations that embody the core aspects of a service-oriented approach to architecture. Each of these implementations extends the basic SOA Reference Model described in this paper.

Component based architecture

A *Component Based Architecture* is an architecture where the functionality of the whole is divided into smaller functions, each *encapsulated* in a component. A *Distributed System* is an extension of components-based architecture and refers to components that may exist in different physical locations.

A simple example of a distributed, component-based architecture is email architecture. Desktop clients, a DNS[9] service and mail servers all interact with each other but are often in different physical locations. This architecture also qualifies as an implementation of SOA.

Figure 1: Distributed system example


User A - Types email and hits "SEND"
Desktop sends "Service request" to eMail server to send an email
DNS Service
Wireless router connects to Email server and sends "Service Request" to get email.
Wireless laptop forwards service request to router
User B - sends "Service Request" to "get" email.
WWW
eMail server accepts incoming "Service request" to forward email to recipient. A Service Request is sent to recipient's email server as a result.
Email server accepts Incoming "Service Request" to forward email to recipient.

Distributed and component-based systems are commonplace in today's enterprise infrastructures. The distributed system in Figure 1 could even show the subcomponents within each major component. For instance, a mail server has an SMTP daemon, a persistent data store, an authentication subsystem (to validate users) and much more.

The main advantage of a component-based architecture is that it facilitates reusability and repurposing of specific components and that it makes maintenance easier. Reusability and repurposing are often primary business drivers for adopting an SOA.

Localized definition for SOA
Several implemented architectures and infrastructures claim to be SOA. In the absence of a formal definition and reference model, amalgamating the principle elements of these implementations captures the main SOA concepts.

The following main concepts are consistent in all SOA implementations:

- Services
- Service descriptions (including security parameters and constraints, reusability and repurpose-ability)
- Advertising and discovery
- Specification of an associated data model
- Service contract

DEFINITION
(Pending the completion of an SOA reference model by a standards body)
Service Oriented Architecture is an architectural paradigm for components of a system and interactions or patterns between them. A component offers a service that waits in a state of readiness. Other components may invoke the service in compliance with a service contract.

Main concepts of SOA
This section explores the main concepts of SOA and provides specific examples of their implementation.

Currently, there is no normative, standards-based consensus on what constitutes SOA. It is the authors' opinion that such a definition should not constrain SOA to adhering to specific standards. The following sections only express the current views of the authors and are not definitive.

Services
A service is a contractually defined behavior that can be implemented and provided by a component for use by another component.

Known implementations
The term "services" does not imply web services; although, web services are well known implementations of services. Other specialized implementations include J2EE[6] and .NET[7].

Service descriptions
The service description consists of the technical parameters, constraints and policies that define the terms to invoke the service. Each service should include a service definition in a standardized format. This enables applications and human actors to examine the service description and determine issues such as, what the service does, how they may bind to it, and what security protocols (if any) must be used with it.

The declaration may also include details about any implied process or other legal or business terms that occur when the service is invoked. For example, if a service consumer invokes a service that places a purchase order to the service provider, and the execution is successful, it may result in a financial responsibility to the service provider or some other legal entity.

Known implementations
While the nature of the services themselves may vary, a common standard for declaring a service is desirable when building an infrastructure. Two such standards exist today: W3C®'s[10] Web Services Description Language[11] (WSDL) and ebXML's Collaboration Protocol Profile[12].

Version 2.0 of WSDL is impressive in its completeness and ease of implementation; however, it only covers the basic aspects of service description.

Individual adopters should carefully consider their own requirements before choosing a standard to adopt in their infrastructure. Consult the notes at the end of this paper for current information about evolving standards. The authors do not endorse any particular standard.

ebXML is a joint SOA initiative between UN/CEFACT[13] and OASIS[14]. In addition to providing technical components, the Collaboration Protocol Profile was developed to meet the specific needs of electronic business that involve service-oriented interactions between legal enterprises.

Advertising and discovery of services
What is advertising?
A service must communicate its service description in an accessible manner to potential consumers. It does so by using one of several advertising methodologies, such as *Pull* and *Push*.

In the *Pull* methodology, potential service consumers request the service provider to send them the service description. This pull methodology may be invoked as a service itself.

In the *Push* methodology, the service provider, or its agent, sends the service description to potential service consumers.

The push and pull methodologies may work together to facilitate advertising services through a third party in a publish-subscribe pattern.

Different models for the push methodology include:

- *Unicast* – Unicast (point-to-point) is a methodology where the service provider sends a message from a single source to a single destination.
- *Multicast* – Multicast (point-to-multipoint) is a parallel communication pattern in which a source host sends a message to a group of destination hosts. This is different from sending multiple, serial, unicast (point-to-point) messages to each of the destination hosts.
- *Broadcast* – Broadcast (point-to-all points) is a methodology where the service provider sends a transmission to all message consumers on a fabric.
- *Anycast* – Anycast (point-to-point-to-multipoint) is a methodology that assigns a private address to several message consumers on a fabric. The message sender does not know or care who consumes the message or the details of the message's distribution list.

If an SOA infrastructure is intended to scale beyond a few services, then it is desirable to implement a neutral, third-party agent specifically for advertising and discovering services.

What is discovery?
Discovery occurs when a potential consumer obtains information about the existence of a service, its applicable parameters and terms. Discovery does not constitute authorization to execute against the service; although these details may be included in the discovery pattern.

Implementing advertising and discovery
The advertising and discovery components of SOA may be implemented in many ways, including using a registry/repository or a services directory. Although using these may make discovery easier, an SOA requires neither of them.

Registry/repository
A Registry/Repository is a component where users can store and manage artifacts required for their enterprise to function. This includes artifacts that require sharing among more than one user (such as XML schemas and web-service descriptions). The repository component provides an intrinsic storage mechanism that is bound to the registry such that the registry knows about any auditable events to the artifacts in the repository.

Services directory
A Directory is an interface that provides information to bind to artifacts. Those who own or control the artifacts may make an entry into the directory to reference the artifact and explain how to bind to it. Others may retrieve this information and use it to bind to the artifacts. The main drawback of the directory is that it has no control or notification if an artifact is deleted, changed or replaced, and the directory may not be able to indicate these events to users.

Both Registry/Repository and Directory implementations allow for federation (also called replication). This functionality allows content from one implementation to be replicated or referenced from within other implementations.

Known implementations
Several standards exist to constrain Registry/Repository and Directory implementations. The most prevalent standards are ISO/IEC 11179 Part 3[15] (an ISO standard for metadata registries), the OASIS ebXML Registry-Repository Technical Specifications[16] and the OASIS Universal Description and Discovery Interface (UDDI) Technical Specification[17].

Standards such as Bluetooth™ [18] use a broadcast-type methodology to advertise their services to other Bluetooth devices that are within range.

Specification of associated data model
When invoking a service, certain parameters may be required to help the service fulfill the service request. The service may also pass parameters back to the service consumer.

To understand any required parameters serialization, an artifact is required to specify the associated data models for the services. Even if no parameters are used, SOA requires a base component to declare this in a format that is understandable to service consumers.

Known implementations
The W3C's WSDL[10] can be used to express that related schema fragments constrain XML instance data being passed in and out of services.

ebXML's Collaboration Protocol Profile[11] also references a schema for instance data being used in a service or collaboration of services.

Both of these implementations are not specifically dependent on the W3C's XML Schema[19] format; yet most implementations use it.

SOA framework

SOA infrastructure

Figure 2 to Figure 4 are concept maps that show the basic concepts of SOA, independent of semantics or technology. Concept maps are informal, graphical representations of concepts and the relationships among them. Figure 2 shows an example of a concept map.

Figure 2: Concept map example

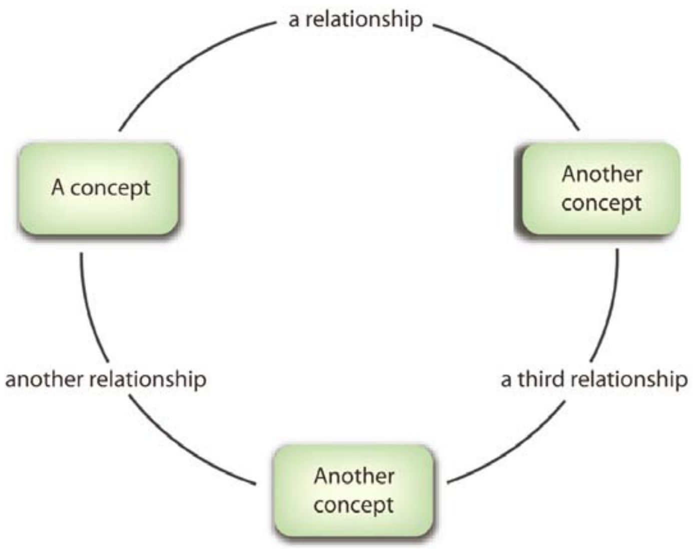
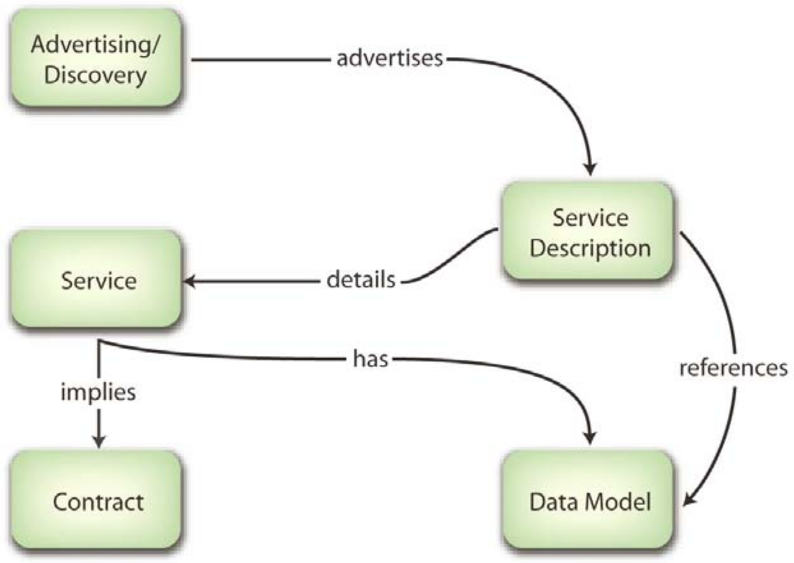


Figure 3 shows a concept drawing of a basic, SOA reference model.

Figure 3: Basic Service Oriented Architecture reference model



Most architectures that are called SOA include a service provider, a service consumer, and some messaging infrastructure. The authors believe that not all of these concepts need to be included to call something an SOA. The service consumer and the contract between the service provider and service consumer are implied. A software architect that designs a software application using all the minimal concepts of SOA has designed an application that is compliant with an SOA.

Optional concepts and shared SOA infrastructure
To fulfill their mandate at runtime, most SOAs may include additional concepts to those shown in Figure 3. The additional concepts are a service consumer, a service client, acceptance of the service contract and invoking the service.

Based on their current understanding of what an SOA reference model can be, the authors believe that it should not be necessary to include optional concept to comply with an SOA reference model. Because no standardized reference model exists, compliance statements are not binding.

Figure 4: Extended Service Oriented Architecture concepts

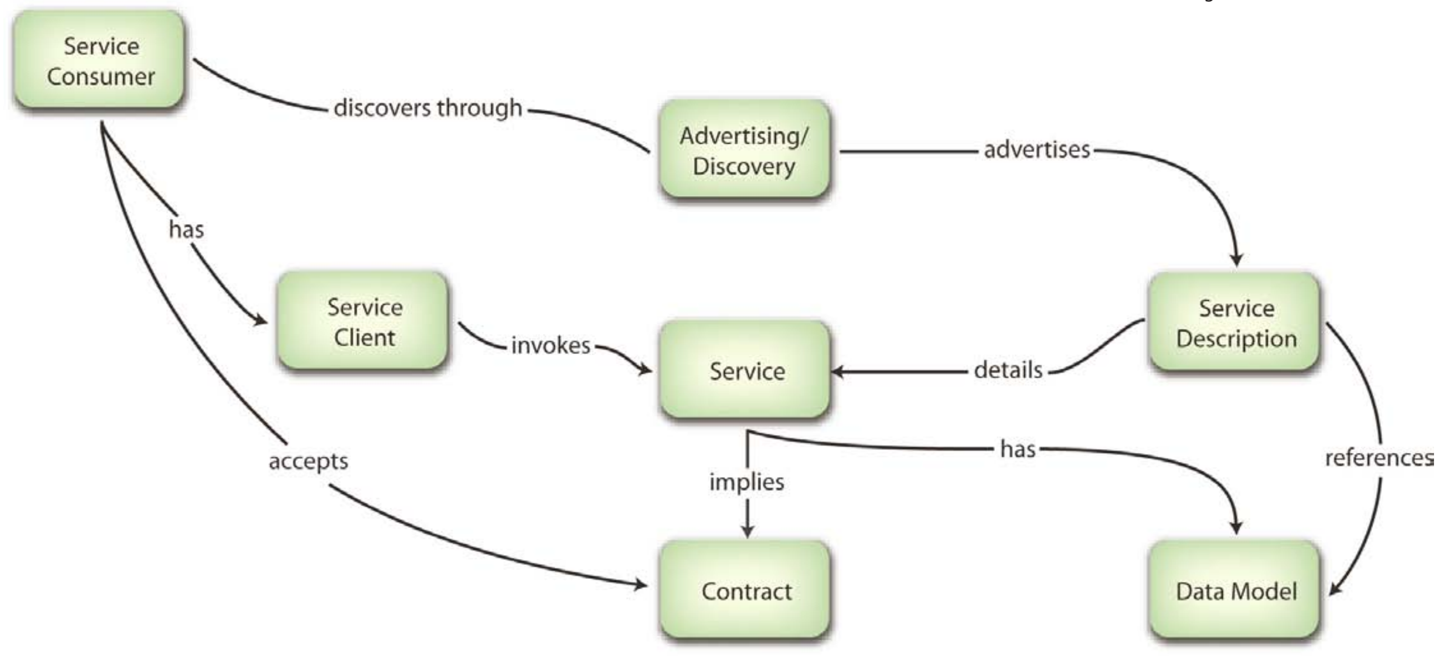


Figure 4 shows optional concepts for an SOA and their interaction with the basic concepts of a SOA.

It is not necessary to invoke a service or have a service consumer to be an SOA. (See Figure 4.) The service provider offers a service, a service contract for its use and a description of the service. The service description references the data model associated with invoking the service.

The service description is advertised using one of many methodologies, such as push (multicast, broadcast or anycast) or pull. The service consumer discovers the service specification through the advertisement mechanism. Discovery does not constitute acceptance of the contract, nor does it imply service invocation; it is merely the action of discovering information about the service. It does not necessarily happen in one action and may require a number of actions. It may also take place by non-electronic means.

Figure 4 excludes other optional specialized functionality, such as security, and avoids calling out specific implementations such as WSDL, XML[20], SOAP[21] or ebXML messaging.

SOA patterns

Figure 5 shows a simplistic service pattern, where a service provider offers a service and a service consumer uses the service. Several types of communication protocols may be used to communicate the request-response pair, and a variety of methodologies, such as *synchronous* or *asynchronous*, may be used. SOA is not bound to any specific communication protocol.

DEFINITION
A **pattern** is a recurring solution to a recurring problem within a specific context. The work is based largely on the architectural methods of Christopher Alexander and has been refined by the Gang of Four to apply specifically to software development.

Figure 5: Basic pattern of Service Oriented Architecture

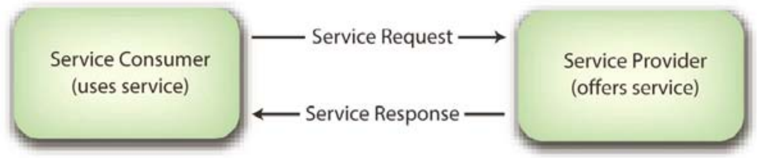

Service Consumer (uses service)
Service Request
Service Response
Service Provider (offers service)

Figure 5 shows the "Request – Response" pattern. For a component-based e-mail system, such as the one shown in Figure 1, the same pattern is used for the following interactions:

1. User A makes a request to her local client to send an e-mail.

2. User A's local client forwards this request to the e-mail server.

3. User A's e-mail server requests DNS service from a DNS server to find the physical location of the recipient's e-mail server.

4. User A's e-mail server sends a request to the recipient's e-mail server to send an e-mail.

5. User B sends a request to his local client to get any e-mails sent to him.

6. User B's local e-mail client sends a request via its router to User B's e-mail server to get his e-mail.

A request may also send *Parameters* (constrained by the services' data model), such as when User B sends a request to get his e-mail. For example, the user may be required to send the following information:

1. E-mail username

2. E-mail user password

3. At the transport level, the location of where to send the response

Business drivers for adopting SOA

Information Technology (IT) workers face many challenges, including:

- Limited budgets
- Constantly changing technologies
- Evolving technologies for the same business function. An example of this may be a new version of software (and *Application Programming Interfaces* or *APIs*) for customer relationship management (CRM).
- Business requirements that demand applications and technology silos that need to be integrated with each other.
- Application functionality that must be extended to reach outside an enterprise firewall (the extended enterprise). Examples are an inventory database that must become accessible to customers to determine availability, or a core banking system that must be extended to customers through the Internet.

Core Business Driver
The core business driver for SOA as an integration technology is its ability to lower integration costs by making it possible to rapidly adapt a system's core functionality to be extended and repurposed to multiple consumers.

Many of these factors require system integration. For example, when a core banking system must be accessed and used by customers through the Internet, a demand is placed on the system to expose some of its functionality to the application that drives the Internet access.

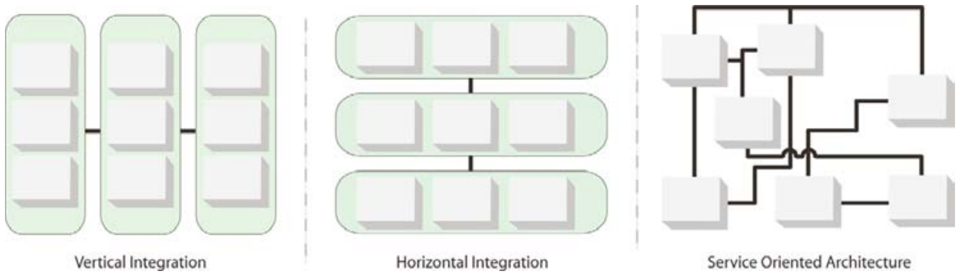
If the integration is always in native format (specific to an application's programming language or environment), then integration projects will require a lot of custom integration work. In addition, the ability to connect multiple systems together quickly will require highly specialized programming capabilities and knowledge of the nuances of the individual systems.

Because each enterprise application may be implemented on a single server that provides the same functionality to many clients, it makes sense to use common protocols to access the functionality. IT staff can lower costs dramatically by using common protocols and standards, and common methodologies such as SOA.

Figure 6 shows the distinct philosophies of each step that business has taken to adapt to requirements integration:

1. Vertical silos of integration – keeping all applications and systems with similar functionality integrated with each other, but not accounting for applications that may wish to use their core functionality in the future.
2. Horizontal integration – integration of some but not all similar functionality across vertical systems; for example using a common purchasing system for raw materials, shipping needs and office supplies.
3. The SOA – an environment of ubiquitous service providers and service consumers interoperating with each other in a secure and consistent manner.

Figure 6: Business views of system integration



Many IT workers are placing requirements for this type of integration on their software vendors. Most of the large and medium-sized software vendors have either announced or incorporated SOA methodology in their software. While the basic patterns of integration remain the same, the specific technology to implement it does vary, depending largely on the software vendor.

While the technical barriers to integration are easy to overcome, there are vastly differing business policies and legal aspects to be enforced at runtime, but no widely accepted standards exist for them yet. Many software vendors continue to pursue different methodologies and individual solutions for these components of the SOA. Many also disagree on the views of the SOA technical stack functionality, or have proprietary perceptions of how SOA should be defined.

In addition to the disagreement on technical standards, a larger problem exists. Many do not share the same views of SOA. IT workers in different businesses have different requirements, and their ideas for solutions vary.

Until a stable and widely accepted reference model for SOA exists, these problems are likely to continue. The areas that will lead integration and consolidation of technologies will likely continue to develop in key vertical areas of focus, with much effort dedicated to solving specialized sets of problems.

This will likely affect several international issues, including:

- The possibility of an outbreak of a fast-spreading contagious disease[24]
- Security risks associated with balancing trade facilitation (and the information-sharing requirements) with global terrorism threats[25]

Hopefully, disagreements will be resolved and finding a multi-vendor solution will take priority over individual interests.

Recently, OASIS started to define SOA independent of any specific technology, software platform, operating system or other environmental variable. The initial deliverable is an SOA reference model.

The consolidation of portions of SOA into common and ubiquitous software will also affect the business view of SOA. The ongoing business requirement for integration and software market forces will affect how organizations invest in developing SOA.

Technical evolution of SOA

The development of SOA can be traced to the era when building monolithic applications was popular, when the first large-scale systems were deployed by businesses and other enterprises. The evolution from *procedural*, or unstructured code, to *Object Oriented*, or structured code, was the first step in the evolution to a modern SOA.

EVOLUTIONARY STEP	EXAMPLE
Monolithic	Large scale application using a procedural coding methodology
Structured or Object Oriented	Dividing applications into units of logic based on functionality. The first steps of SOA.
Clients and Servers	The logical progression of OO – bundling groups of functions on one computer (server) and invoking them from another (client).
3 Tier	Adding an extra layer to interaction. The primary driver was to create an interface that was agnostic to the specific environment of the server. For example, if you make an HTTP request to a server, the middle tier translates your HTTP get() request into the native format that is needed to get the resource requested.
Nth Tier	Layered request-response calls between applications. Portal development relied on this concept.
Distributed Objects	A ubiquitous and heterogeneous system of many distributed, orthogonal objects rather than a simple, 2 or 3 computer interaction.
Components	Aggregating objects into logical components that achieve specific functionality (often mapping to a component or the builder's requirements) and creating interfaces to those components. An example is a database server used as a persistent data-storage component for CRM software.
Service Oriented Components	A ubiquitous environment of orthogonal components interacting in a peer-based environment, often using service provider proxies (interfaces based on widely accepted standards) to offer services.

Conclusions

SOA is based on the use of distributed objects and components and is the next evolutionary step in computing environments. SOA does not have a standardized, reference model yet; however, implementations share the main concepts of services, service descriptions, advertising and discovery, the specification of an associated data model, and the use of a service contract. An SOA may also implement optional concepts that include a service consumer, a service client, acceptance of the service contract and invoking the service.

There are many business drivers affecting the development of a standardized SOA reference model. Once this is achieved, SOA will likely be part of the solution for many business and world issues.

References

[1] Web Services are generally referred to as a group of loosely organized specifications. No formal or normative definition of scope exists to define web services.

[2] ebXML is an initiative started by OASIS and UN/CEFACT to build a global infrastructure for electronic business. It exists as a set of related specifications. Many of the specifications are now part of ISO. See *www.ebxml.org*

[3] Organization for the Advancement of Structured Information Systems (OASIS), Service Oriented Architecture Reference Model Technical Committee (SOA-RM TC), *www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm*

[4] A component-based architecture is the model of separating functionality into individual, freestanding objects that can work together.

[5] Distributed Component Object Model - *www.microsoft.com/com/default.mspx*

[6] CORBA is the Object Management Group (OMG) Component Object Reference Broker Architecture. IT is a vendor-neutral, component-based architecture. *www.omg.org/gettingstarted/corbafaq.htm*

[7] Sun Microsystems® Java 2® Platform, Enterprise Edition (J2EE) defines the standard for developing component-based, multi-tier, enterprise applications. *http://java.sun.com/j2ee/*

[8] Microsoft® .NET is a set of Microsoft software technologies for connecting information, people, systems, and devices. *www.microsoft.com/net/*

[9] Domain Name Service - Used to map names to and from IP addresses.

[10] The W3C Consortium is a nonprofit, global, standards body that develops Internet-related standards. The official website of the W3C is at *www.w3c.org*

[11] Details of WSDL can be found at the W3C's website at *www.w3.org/2002/ws/desc/*

[12] The ebXML CPP or Collaboration Protocol Profile work is at *www.oasis-open.org/committees/tc_home.php?wg_abbrev=ebxml-cppa*

[13] The United Nations Centre for Facilitation of Commerce and Trade (UN/CEFACT) is a globally sanctioned, UN body dedicated to developing trade facilitation and electronic business standards and technologies to be used royalty free by anyone, worldwide. *www.unece.org/cefact/*

[14] The Organization for the Advancement of Structured Information Systems (OASIS) is a nonprofit, standards-development organization. *www.oasis-open.org*

[15] ISO/IEC 11179 Part 3 is available from the ISO website at: *www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=39488&scopelist=*

[16] There are two relevant OASIS ebXML Registry and Repository specifications. The Registry Services Specification (RSS) and the Registry Information Model are available from: *www.oasis-open.org/committees/tc_home.php?wg_abbrev=regrep*

[17] The OASIS UDDI specification is available from the OASIS website under the technical committee's home page area at *www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec*

[18] Bluetooth is a short range, wireless-radio standard for broadcasting (multi casting) - *www.bluetooth.com/*

[19] XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a way to define the structure, content and semantics of XML documents. *www.w3.org/XML/Schema*

[20] The eXtensible Markup Language is a W3C protocol. It is a text format for serialization of data. *www.w3.org/XML/*

[21] The W3C's Simple Object Access Protocol is an XML based messaging format. *www.w3.org/TR/soap/*

[22] Christopher Alexander is Professor in the Graduate School and Emeritus Professor of Architecture at the University of California, Berkeley. He is the father of the Pattern Language movement in computer science, and A Pattern Language, a seminal work and possibly the first complete book ever written in hypertext. *www.patternlanguage.com/leveltwo/ca.htm*

[23] The Gang of Four consists of Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *http://hillside.net/patterns/DPBook/GOF.html*

[24] The Center for Disease control has investigated integrating medical and health systems worldwide in order to track and detect the outbreak of rapidly infectious contagions. A prototype was developed for the XML 2003 conference in Philadelphia, U.S.A. by a team that included Adobe Systems® and Sun Microsystems.

[25] The World Customs Organization (WCO) has introduced a U.S. led initiative after the tragic events of September 11, 2001. Called the Secure Container Initiative (SCI), it uses an SOA methodology to ensure information about shipped goods reaches the destination before the goods do, so that methodologies can be applied to assess security risks.

Service Oriented Architecture Whitepaper

Author: Duane Nickull – Adobe Systems, Inc.

Contributors: Mike Connor, C. Matthew MacKenzie, Ben Watson, Mark Cowan, Ed Chase – Adobe Systems, Inc.

Adobe Systems Incorporated • 345 Park Avenue, San Jose, CA 95110-2704 USA • *www.adobe.com*

Adobe®