Service-oriented Architecture (SOA) defined

Source: Wikipedia.org and XML.com, 15 Oct 2007, edited by Raymond Kurland, TechniCom Group.

Our editing: We have edited the articles and removed all of the hyperlinks to further explanations to make this document more readable. Go to wikipedia.org and search on "service oriented architecture" to explore the hyperlinks. Ray can be contacted at rayk@technicom.com. Visit www.cad-portal.com for other interesting articles on the PLM Industry and www.technicom.com for more information about the TechniCom Group LLC.

Service-oriented Architecture (SOA) is an architectural design pattern that concerns itself with defining loosely-coupled relationships between producers and consumers. While it has no direct relationship with software, programming, or technology, it's often confused with an evolution of distributed computing and modular programming.

There is no widely agreed upon definition of SOA other than its literal translation. It is an architecture that relies on service-orientation as its fundamental design principle. In an SOA environment independent services can be accessed without knowledge of their underlying platform implementation. These concepts can be applied to business, software and other types of producer/consumer systems.

Service-oriented architecture

Every business comprises core and non-core functions. The core functionality doesn't change very frequently and the non-core changes very frequently. For example, the retail store will always sell goods and this will be one of the core functions, but the way it will sell the goods might differ with time and market needs, etc. These are the non-core functions which change very frequently.

In a software industry, it is desirable that the functions that change frequently should be decoupled from functions that change infrequently.

In a simplistic term, SOA is the practice of segregating the core business functions into independent services that don’t change frequently. Going further it also extends this segregation to many things that can logically and functionally be separated, regardless of whether they’re changeable or not.

Service-oriented architecture (SOA) is an architectural style where existing or new functionalities are grouped into atomic services. These services communicate with each other by passing data from one service to another, or by coordinating an activity between one or more services.

Companies have long sought to integrate existing systems in order to implement information technology (IT) support for business processes that cover all present and prospective systems requirements needed to run the business end-to-end. A variety of designs can be used to this end, ranging from rigid point-to-point electronic data interchange (EDI) interactions to Web auctions. By updating older technologies, such as Internet-enabling EDI-based systems, companies can make their IT systems available to internal or external customers; but the resulting systems have not proven to be flexible enough to meet business demands.

A flexible, standardized architecture is required to better support the connection of various applications and the sharing of data. SOA is one such architecture. It unifies business processes by structuring large applications as an ad-hoc collection of

smaller modules called services. These applications can be used by different groups of people both inside and outside the company, and new applications built from a mix of services from the global pool exhibit greater flexibility and uniformity. One should not, for example, have to provide redundantly the same personal information to open an online checking, savings or IRA account, and further, the interfaces one interacts with should have the same look and feel and use the same level and type of input data validation. Building all applications from the same pool of services makes achieving this goal much easier and more deployable to affiliate companies. An example of this might be interacting with a rental car company's reservation system even though you are doing so from an airline's reservation system.

Requirements for a SOA

In order to efficiently use a SOA, one must meet the following requirements:

- Interoperability between different systems and programming languages - The most important basis for a simple integration between applications on different platforms is a communication protocol, which is available for most systems and programming languages.
- Clear and unambiguous description language - To use a service offered by a provider, it is not only necessary to be able to access the provider system, but the syntax of the service interface must also be clearly defined in a platform-independent fashion.
- Retrieval of the service - To allow a convenient integration at design time or even system run time, a search mechanism is required to retrieve suitable services. The services should be classified as computer-accessible, hierarchical or taxonomies based on what the services in each category do and how they can be invoked.

Webservices approach to a service-oriented architecture

Web services could be used to implement a service-oriented architecture. A major focus of Web services is to make functional building blocks accessible over standard Internet protocols that are independent from platforms and programming languages. These services can be new applications or just wrapped around existing legacy systems to make them network-enabled. A service can rely on another service to achieve its goals.

Each SOA building block can play one or more of three roles:

- **Service provider** - The service provider creates a Web service and possibly publishes its interface and access information to the service registry. Each provider must decide which services to expose, how to make trade-offs between security and easy availability, how to price the services, or, if they are free, how to exploit them for other value. The provider also has to decide what category the service should be listed in for a given broker service and what sort of trading partner agreements are required to use the service.
- **Service broker** - The service broker, also known as service registry, is responsible for making the Web service interface and implementation access information available to any potential service requestor. The implementer of the broker decides about the scope of the broker. Public brokers are available through the Internet, while private brokers are only accessible to a limited audience, for example, users of a company intranet. Furthermore, the amount of the offered information has to be

decided. Some brokers specialize in many listings. Others offer high levels of trust in the listed services. Some cover a broad landscape of services and others focus within an industry. There are also brokers that catalog other brokers. Depending on the business model, brokers can attempt to maximize look-up requests, number of listings or accuracy of the listings. The Universal Description Discovery and Integration (UDDI) specification defines a way to publish and discover information about Web services.

- **Service requestor** - The service requestor or Web service client locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke one of its Web services.

Other SOA Concepts

It is not true that the Architecture is not tied to a specific technology. Those who say that it may be implemented using a wide range of technologies, including EJB, EMS, RPC, DCOM, CORBA, DDS or WCF attest to not having grasped the revolutionary meaning of Service Oriented Architecture: none of the technologies mentioned above are able to implement self-describing entities as is possible using Web Services. By using the Web Service Definition Language WSDL it is possible that a client can be completely "agnostic" about a service, gaining "understanding" during run-time of all the semantics exposed in a service without "knowing" "a priori" protocols, binding, data types, policies, Service Level Agreement SLA. This means that the client can be almost deprived of application logic, thus evolving the "client-server" approach towards the new Service Oriented approach: an example of this new frontier is well demonstrated, for instance, by the adoption of SOA in the implementation of Network Centric Warfare environments.

SOA can be implemented using one or more of protocols and, for example, might use a file system mechanism to communicate data conforming to a defined interface specification between processes. The key is independent services with defined interfaces that can be called to perform their tasks in a standard way, without the service having foreknowledge of the calling application, and without the application having or needing knowledge of how the service actually performs its tasks.

SOA can also be regarded as a style of information systems architecture that enables the creation of applications that are built by combining loosely coupled and interoperable services. These services inter-operate based on a formal definition (or contract, e.g., WSDL) that is independent of the underlying platform and programming language. The interface definition hides the implementation of the language-specific service. SOA-based systems can therefore be independent of development technologies and platforms (such as Java, .NET etc). Services written in C# running on .NET platforms and services written in Java running on Java EE platforms, for example, can both be consumed by a common composite application (or client). Applications running on either platform can also consume services running on another, being Web services, which facilitate reuse. Many COBOL legacy systems can also be wrapped by a managed environment and to be presented as a software service. This has allowed the useful life of many core legacy systems to be extended indefinitely no matter what language they were originally written in.

SOA can support integration and consolidation activities within complex enterprise systems, but SOA does not specify or provide a methodology or framework for documenting capabilities or services.

High-level languages such as BPEL and specifications such as WS-CDL and WS-Coordination extend the service concept by providing a method of defining and supporting orchestration of fine grained services into more coarse-grained business services, which in turn can be incorporated into workflows and business processes implemented in composite applications or portals.

The use of Service component architecture (SCA) to implement SOA is a current area of research.

SOA definitions

SOA is a design for linking business and computational resources (principally organizations, applications and data) on demand to achieve the desired results for service consumers (which can be end users or other services). OASIS (the Organization for the Advancement of Structured Information Standards) defines SOA as the following:

A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

There are multiple definitions of SOA, the OASIS group and the Open Group have created formal definition with depth which can be applied to both the technology and business domains. We like the description provided by XML.com:

- Service-Oriented Architecture (XML.com) - SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software agents on behalf of their owners. How does SOA achieve loose coupling among interacting software agents? It does so by employing two architectural constraints: First, a small set of simple and ubiquitous interfaces to all participating software agents. Only generic semantics are encoded at the interfaces. The interfaces should be universally available for all providers and consumers. Second, descriptive messages constrained by an extensible schema delivered through the interfaces. No, or only minimal, system behavior is prescribed by messages. A schema limits the vocabulary and structure of messages. An extensible schema allows new versions of services to be introduced without breaking existing services.
- Other definitions exist by several groups, but all describe similar approaches.

Though many definitions of SOA limit themselves to technology or just web services, this is predominantly pushed by technology vendors; in 2003 they talked just of web services, while in 2006 the talk was of events and process engines.

Why SOA?

Enterprise architects believe that SOA can help businesses respond more quickly and cost-effectively to changing market conditions[5]. This style of architecture promotes reuse at the macro (service) level rather than micro (classes) level. It can also simplify interconnection to - and usage of - existing IT (legacy) assets.

In some respects, SOA can be considered an architectural evolution rather than a revolution and captures many of the best practices of previous software architectures. In communications systems, for example, there has been little development of solutions that use truly static bindings to talk to other equipment in

the network. By formally embracing a SOA approach, such systems are better positioned to stress the importance of well-defined, highly inter-operable interfaces.

Some have questioned whether SOA is just a revival of modular programming (1970s), event-oriented design (1980s) or interface/component-based design (1990s). SOA promotes the goal of separating users (consumers) from the service implementations. Services can therefore be run on various distributed platforms and be accessed across networks. This can also maximize reuse of services.

SOA and web service protocols

SOA may be built on Web services standards (e.g., using SOAP) that have gained broad industry acceptance. These standards (also referred to as web service specifications) also provide greater interoperability and some protection from lock-in to proprietary vendor software. One can, however, implement SOA using any service-based technology, such as Jini.

Service-oriented architecture is often defined as services exposed using the Web Services Protocol Stack. The base level of web services standards relevant to SOA includes the following:

- XML - a markup language for describing data in message payloads in a document format
- HTTP (or HTTPS) - request/response protocol between clients and servers used to transfer or convey information
- SOAP - a protocol for exchanging XML-based messages over a computer network, normally using HTTP
- XACML - a markup language for expressing access control rules and policies.
- Web Services Description Language (WSDL) - XML-based service description that describes the public interface, protocol bindings and message formats required to interact with a web service
- Universal Description, Discovery, and Integration (UDDI) - An XML-based registry to publish service descriptions (WSDL) and allow their discovery
- Note, however, that a system does not necessarily need to use any or all of these standards to be "service-oriented." For example, some service oriented systems have been implemented using Corba, Jini and REST.

SOA, Web 2.0, and mashups

Web 2.0 refers to a "second generation" of web sites, primarily distinguished by the ability of visitors to contribute information for collaboration and sharing. Web 2.0 applications use Web services and may include Ajax, Flash, Silverlight or JavaFX user interfaces, Web syndication, blogs, and wikis. While there are no set standards for Web 2.0, it is characterized by building on the existing web server architecture and using services. Web 2.0 can therefore be regarded as displaying some SOA characteristics[9].

Mashups are also regarded by some as Web 2.0 applications. The term "enterprise mashup" has been coined to describe Web applications that combine content from more than one source into an integrated experience, which share many of the characteristics of service-oriented business applications (SOBAs), which are applications composed of services in a declarative manner. There is ongoing debate

about "the collision of Web 2.0, mashups, and SOA", with some stating that Web 2.0 applications are a realisation of SOA composite and business applications. [10]

SOA 2.0 or Advanced SOA

Amid much negative reaction, Oracle is taking up SOA 2.0 as "the next-generation version of SOA" combining service-oriented architecture and Event Driven Architecture, and categorizing the first iteration of SOA as client-server driven. Even though Oracle indicates that Gartner is coining a new term, Gartner analysts indicate that they call this advanced SOA and it is 'whimsically' referred to as SOA 2.0.[12] Most of the pure-play middleware vendors (e.g., webMethods and TIBCO Software) have had SOA 2.0 attributes for years. SOA 2.0 can therefore be regarded as "more marketing noise than anything else" and product evangelism rather than a new "way of doing things".

However, other industry commentators have criticized attaching a version number to an application architecture design approach, while others have stated that the "next generation" should apply to the evolution of SOA techniques from IT optimization to business development.

What are the challenges faced in SOA adoption?

One obvious and common challenge faced is managing **services metadata**. SOA-based environments can include many services which exchange messages to perform tasks. Depending on the design, a single application may generate millions of messages. Managing and providing information on how services interact is a complicated task.

Another challenge is providing **appropriate levels of security**. Security model built into an application may no longer be appropriate when the capabilities of the application are exposed as services that can be used by other applications. That is, application-managed security is not the right model for securing services. A number of new technologies and standards are emerging to provide more appropriate models for security in SOA. See SOA Security entry for more info.

As SOA and the WS-* specifications are constantly being expanded, updated and refined, there is a shortage of skilled people to work on SOA based systems, including the integration of services and construction of services infrastructure.

Interoperability is another important aspect in the SOA implementations. The WS-I organization has developed Basic Profile (BP) and Basic Security Profile (BSP) to enforce compatibility. Testing tools have been designed by WS-I to help assess whether web services are conformant with WS-I profile guidelines. Additionally, another Charter has been established to work on the Reliable Secure Profile.

There is significant **vendor hype** concerning SOA that can create expectations that may not be fulfilled. Product stacks are still evolving as early adopters test the development and runtime products with real world problems. SOA does not guarantee reduced IT costs, improved systems agility or faster time to market. Successful SOA implementations may realise some or all of these benefits depending on the quality and relevance of the system architecture and design.

Criticisms of SOA

Some criticisms of SOA are based on the assumption that SOA is just another term for Web Services. For example, some critics claim SOA results in the addition of XML layers introducing XML parsing and composition. In the absence of native or binary forms of Remote Procedure Call (RPC) applications could run slower and require

more processing power, increasing costs. Most implementations do incur these overheads, but SOA can be implemented using technologies (for example, Java Business Integration (JBI)) which do not depend on remote procedure calls or translation through XML. However, there are emerging and open-source XML parsing technologies, such as VTD-XML, and various XML-compatible binary formats (http://vtd-xml.sf.net/persistence.html) that promise to significantly improve the SOA performance.

Stateful services require both the consumer and the provider to share the same consumer-specific context, which is either included in or referenced by messages exchanged between the provider and the consumer. The drawback of this constraint is that it could reduce the overall scalability of the service provider because it might need to remember the shared context for each consumer. It also increases the coupling between a service provider and a consumer and makes switching service providers more difficult.

Another concern is that WS-* standards and products are still evolving (e.g., transaction, security), and SOA can thus introduce new risks unless properly managed and estimated with additional budget and contingency for additional Proof of Concept work.

An informal survey by Network Computing placed SOA as the most despised buzzword (November 2006).

Some critics feel SOA is merely an obvious evolution of currently well-deployed architectures (open interfaces, etc).

A SOA being an architecture is the first stage of representing the system components that interconnect for the benefit of the business. At this level a SOA is just an evolution of an existing architecture and business functions. SOAs are normally associated with interconnecting back end transactional systems that are accessed via web services.

The real issue with any IT "architecture" is how one defines the information management model and operations around it that deal with information privacy, reflect the business's products and services, enable services to be delivered to the customers, allow for self care, preferences and entitlements and at the same time embrace identity management and agility. On this last point, system modification (agility) is a critical issue which is normally omitted from IT system design. Many systems, including SOAs, hard code the operations, goods and services of the organization thus restricting their online service and business agility in the global market place.

Adopting SOAs is therefore just the first (diagrammatic) step in defining a real business system. The next step in the design process is the definition of a Service Delivery Platform (SDP) and its implementation. It is in the SDP design phase where one defines the business information models, identity management, products, content, devices, and the end user service characteristics, as well as how agile the system is so that it can deal with the evolution of the business and its customers.

SOA and Business Architecture

One area where SOA has been gaining ground is in its power as a mechanism for defining business services and operating models and thus provide a structure for IT to deliver against the actual business requirements and adapt in a similar way to the business. The purpose of using SOA as a business mapping tool is to ensure that the services created properly represent the business view and are not just what

technologists think the business services should be. At the heart of SOA planning is the process of defining architectures for the use of information in support of the business, and the plan for implementing those architectures (Enterprise Architecture Planning by Steven Spewak and Steven Hill). Enterprise Business Architecture should always represent the highest and most dominant architecture. Every service should be created with the intent to bring value to the business in some way and must be traceable back to the business architecture.

Within this area, SOMA (Service-Oriented Modeling and Architecture) was announced by IBM as the first SOA-related methodology in 2004. Since then, efforts have been made to move towards greater standardization and the involvement of business objectives, particularly within the OASIS standards group and specifically the SOA Adoption Blueprints group. All of these approaches take a fundamentally structured approach to SOA, focusing on the Services and Architecture elements and leaving implementation to the more technically focused standards.

SOA and network management architecture

Tools for managing SOA infrastructure include:

- Symantec APM
- HyPerformix IPS Performance Optimizer
- HP Management Software / Mercury SOA Manager
- IBM Tivoli Framework
- Tidal Software Intersperse

Jargon

SOA is an architectural style rather than a product. Several vendors offer products which can form the basis of, or enable, SOA--particularly Enterprise Service Bus (ESB) products. ESBs provide infrastructure that can be purchased, implemented and leveraged for SOA-based systems. SOA relies heavily on metadata design and management. Metadata design and management products are also critical to implementing SOA architectures. See the list of SOA related products for an overview and ideas.