SOA (Service-Oriented Architecture)

By: **IBM Cloud Education**

7 April 2021

Integration

What is SOA, or service-oriented architecture?

SOA (Service-Oriented Architecture)

Explore SOA (service-oriented architecture), an important stage in the evolution of application development and integration.

What is SOA, or service-oriented architecture?

SOA, or service-oriented architecture, defines a way to make software components reusable and interoperable via service interfaces. Services use common interface standards and an architectural pattern so they can be rapidly incorporated into new applications. This removes tasks from the application developer who previously redeveloped or duplicated existing functionality or had to know how to connect or provide interoperability with existing functions.

Each service in an SOA embodies the code and *data* required to execute a complete, discrete business function (e.g. checking a customer's credit, calculating a monthly loan payment, or processing a mortgage application). The service interfaces provide loose coupling, meaning they can be called with little or no knowledge of how the *service* is implemented underneath, reducing the dependencies between applications.

This interface is a service contract between the service provider and service consumer. Applications behind the service interface can be written in Java, Microsoft .Net, Cobol or any other programming language, supplied as packaged software applications by a vendor (e.g., SAP), SaaS applications (e.g., Salesforce CRM), or obtained as open source applications. Service interfaces are frequently defined using Web Service Definition Language (WSDL) which is a standard tag structure based on xml (extensible markup language).

The services are exposed using standard network protocols—such as SOAP (simple object access protocol)/HTTP or Restful HTTP (JSON/HTTP)—to send requests to read or change data. Service governance controls the lifecycle for development and at the appropriate stage the services are published in a *registry* that enables developers to quickly find them and reuse them to assemble new applications or business processes.

These services can be built from scratch but are often created by exposing functions from legacy systems of record as service interfaces.

In this way, SOA represents an important stage in the evolution of application development and integration over the last few decades. Before SOA emerged in the

late 1990s, connecting an application to data or functionality housed in another system required complex point-to-point integration—integration that developers had to recreate, in part or whole, for each new development project. Exposing those functions through SOA *services allowed the developer to simply reuse the existing capability and connect through the SOA ESB architecture* (see below).

Note that although SOA, and the more recent microservices architecture, share many words in common(i.e. "service" and "architecture"), they are only loosely related and, in fact, operate at different scopes, as discussed later in this article.

What is an ESB?

An ESB, or enterprise service bus, is an architectural pattern whereby a centralized software component performs integrations between applications. It performs transformations of data models, handles connectivity/messaging, performs routing, converts communication protocols and potentially manages the composition of multiple requests. The ESB can make these integrations and transformations available as a service interface for reuse by new applications. The ESB pattern is typically implemented using a specially designed integration runtime and toolset that ensures the best possible productivity.

It is possible to implement an SOA without an ESB, but this would be equivalent to just having a bunch of services. Each application owner would need to directly connect to any service it needs and perform the necessary data transformations to meet each of the service interfaces. This is a lot of work (even if the interfaces are reusable) and creates a significant maintenance challenges in the future as each connection is point to point. In fact, ESBs were, eventually, considered such a de facto element of any SOA implementation that the two terms are sometimes used as synonyms, creating confusion.

Read more about ESBs

Benefits of SOA

Compared to the architectures that preceded it, SOA offered significant benefits to the enterprise:

- **Greater business agility; faster time to market:** Reusability is key. The efficiency of assembling applications from reusable services - i.e. building blocks, rather than rewriting and reintegrating with every new development project, enables developers to build applications much more quickly in response to new business opportunities. The service oriented architectural approach supports scenarios for application integration, data integration, and service orchestration style automation of business processes or workflows. This speeds software design and software development by enabling developers to spend dramatically less time integrating and much more time *focusing on delivering and* improving their applications.

- **Ability to leverage legacy functionality in new markets:** A well-crafted SOA enables developers to easily take functionality 'locked' in one computing platform or environment and extend it to new environments and markets. For example, many companies have used SOA to expose functionality from mainframe-based financial systems to new web applications, enabling their customers to serve themselves to processes and information previously accessible only through direct interaction with the company's employees or business partners.

- **Improved collaboration between business and IT:** In an SOA, services can be defined in business terms (e.g., 'generate insurance quote' or 'calculate capital equipment ROI'). This enables business analysts to work more effectively with developers on important insights—such as the scope of a business process defined using services or the business implications of changing a process—that can lead to a better result.

SOA Examples

By 2010, SOA implementations were going full steam at leading companies in virtually every industry. For example:

- Delaware Electric turned to SOA to integrate systems that previously did not talk to each other, resulting in development efficiencies that helped the

organization stay solvent during a five-year, state-mandated freeze on electric rates.

- Cisco adopted SOA to make sure its product ordering experience was consistent across all products and channels by exposing ordering processes as services that Cisco's divisions, acquisitions, and business partners could incorporate into their *websites.*

- Independence Blue Cross (IBC) of Philadelphia implemented an SOA to ensure that the different constituents dealing with patient data—IBC customer service agents, physicians' offices, IBC web site users—were working with the same data source (a 'single source of truth').

SOA vs. microservices

Experts have filled a few thousand print and digital pages comparing SOA and microservices, and defining the subtleties of their relationship to one another. For the purposes of this article, the chief differences between the two are the coupling of components and scope of use:

- SOA is an integration architectural style and an enterprise-wide concept. It enables existing applications to be exposed over loosely-coupled interfaces, each corresponding to a business function, that enables applications in one part of an extended enterprise to reuse functionality in other applications.

- Microservices architecture is an application architectural style and an application-scoped concept. It enables the internals of a single application to be broken up into small pieces that can be independently changed, scaled, and administered. It does not define how applications talk to one another—for that we are back to the enterprise scope of the service interfaces provided by SOA.

Microservices architecture emerged and gained steam with the rises of virtualization, cloud computing, Agile development practices, and DevOps. Most of the advantages of microservices in these contexts arise from the decoupling of the components, which simplifies and improves the following:

- **Developer agility and productivity:** Microservices enable developers to incorporate new technologies to one part of the application without affecting

the rest of the application. Any component can be modified, tested, and deployed independently of the others, which speeds iteration cycles.

- **Scalability:** Microservices can take maximum advantage of cloud scalability—any component can be scaled independently of the others for the fastest possible response to workload demands and the most efficient use of computing resources.

- **Resilience:** Again, thanks to decoupling, the failure of one microservice does not impact the others. And each microservice can perform to its own availability requirements without staking the other components or the entire application to greatest common availability requirements.

For a deeper dive into the differences between SOA and microservices, see "SOA vs. Microservices: What's the difference?"

In the same way that microservices architecture has the potential to bring improvements in agility, scalability, and resilience to application design, these same techniques can also be applied to integration. This is important because, over time, the heavily centralized ESB pattern and its associated centralized team of integration specialists can become a bottleneck. Borrowing from microservices principles, we can potentially break up the ESB into a more fine-grained, decentralized integrations. This is one of the core premises behind agile integration.

SOA and IBM Cloud

As your company shifts its IT infrastructure toward a hybrid cloud approach, there's a high likelihood you'll be transforming a variety of workloads, including those based on SOA, to more lightweight and flexible cloud deployment models.

These transformations are just part of the application modernization required as the demand for better customer experiences and more applications impacts your business and IT operations. To meet these demands, a move toward greater automation also helps. Ideally, it would start with small, measurably successful projects that you can then scale and optimize for other processes and in other parts of your organization.

Working with IBM, you'll have access to AI-powered automation capabilities, including prebuilt workflows, to help accelerate innovation by making every process more intelligent.

Take the next step:

- Open new channels of interaction with customers, suppliers and partners with the IBM Cloud Pak for Integration, a hybrid integration solution that provides an automated and closed-loop lifecycle across multiple styles of enterprise integration.
- Learn how you can connect all your applications and data across multiple private and public clouds to create personalized customer experiences by visiting IBM cloud integration.
- Download the IBM Application Modernization Field Guide to learn how to accelerate modernization, improve developer productivity and enhance operational efficiency and standardization.
- Take our integration maturity assessment to evaluate your integration maturity level across critical dimensions and discover the actions you can take to get to the next level.
- Download our agile integration guide, which explores the merits of a container-based, decentralized, microservices-aligned approach for integrating solutions.
- Read about the five "must-have's" for automation success (link resides outside IBM) in this HFS Research report.

Get started with an IBM Cloud account today.

Cloud Paks

Cloud pricing

View all products

View all solutions

Learn about

What is Hybrid Cloud?

What is Cloud Computing?

What is Confidential Computing?

What is a Data Lake?

What is a Data Warehouse?

What is Machine Learning?

What is DevOps?

What is Kubernetes?

What is Microservices?

Resources

Get started

Docs

Architectures

IBM Garage

Training and Certifications

Partners

Cloud blog

Hybrid Cloud careers

My Cloud account

Site feedback

Let’s talk