



What You Need to Know About SOA

At JusticeExperts.com we try to help clients and business partners realize the promise that lies in the use of web services - - and, over time, in moving toward a Service-Oriented Architecture.

This is not an overnight project, but instead an evolution...One that requires some education, some faith, and a paradigm shift from how Justice does business with information.

To begin, JusticeExperts.com establishes some ground rules and education around the basic concepts of SOA:

An SOA Glossary

Enterprise service bus: A software infrastructure that uses a standard interface and messaging to integrate applications; one way to implement an SOA. (Note: The term, which was coined in a report by Gartner, is relatively new.)

Loosely coupled: The use of well-defined interfaces to connect services; SOAs are built using a loosely coupled approach, where a change in one service does not require changes in linked services.

Message-oriented middleware (MOM): Sometimes referred to as a message-oriented architecture, MOM provides a mechanism for connecting various applications, even across platforms. Data resides in message queues where receiving programs can retrieve it without creating a direct connection with the sending applications.

Publish-subscribe: System where services post (or "publish") data that other services can request (or "subscribe" to). When the published information changes, the subscribed services automatically receive updates.

Service-oriented architecture (SOA): An architecture built around a collection of reusable components with well-defined interfaces.



Service-oriented architecture isn't a totally new approach to software design. Some of the notions behind SOA have been around for years. The underlying concepts date back to the early 1970s, when researchers started drawing boundaries around software and providing access to that software only through well-defined interfaces (an idea called encapsulation). But lately, SOA has been gaining traction, especially as enterprises begin to think seriously about Web Services. The Gartner Group estimates that by 2008, more than 60 percent of enterprises will use SOA as a "guiding principle" when creating mission-critical applications and processes.

But to implement an SOA, you must first understand it—and that isn't always easy. So let's begin with some simple questions and (hopefully) simple answers.

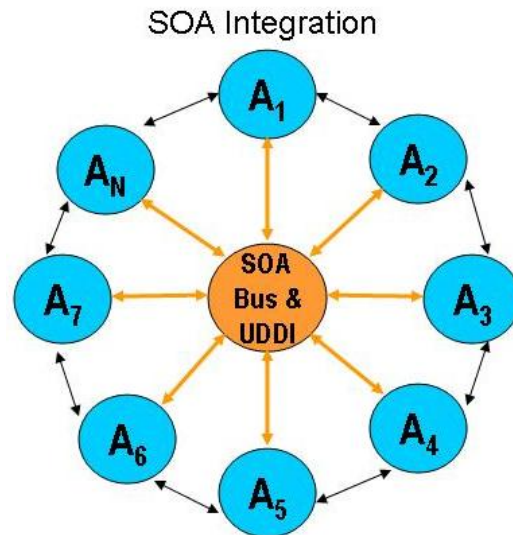
What Is an SOA?

SOAs start with services, which are groups of software components that carry out business processes, for example, verifying a credit card transaction or processing a purchase order. At its most basic, an SOA is a collection of services on a network that communicate with one another. The services are loosely coupled (meaning that an application doesn't have to know the technical details of another application in order to talk to it), have well-defined, platform-independent interfaces, and are reusable. SOA is a higher level of application development (also referred to as coarse granularity) that, by focusing on business processes and using standard interfaces, helps mask the underlying technical complexity of the IT environment. It's like translating a high school science text for your kindergarten-age daughter; you can tell her that the heart pumps blood without getting referring to the mitral valve and pulmonary veins.

Is SOA the new Middleware?

No. SOA is an evolution from traditional tightly coupled application connections—including common object request broker architecture, or CORBA—to loosely coupled ones, such as Web services. Tight coupling makes it hard for applications to adapt to changing business requirements, as each modification to one application may force developers to make changes in other connected applications. Also, object-oriented development uses a finer level of granularity—objects might be defined at the level of employee or customer order. In an SOA, a service is defined at a more abstract level, say, a business process such as creating a new arrest or issuing a warrant.





What Are the Benefits of Adopting an SOA?

SOAs make it easier to integrate the heterogeneous IT environments found in most agencies and enterprises. That's the big value of an SOA; it works very well in mismatched technical environments. Developers don't have to spend an inordinate amount of time writing new lines of code to connect applications. Instead, they can use standard protocols, such as Web services. And large chunks of SOA code are reusable, reducing development costs. An SOA takes the existing legacy investments—your Mainframe systems, IBM, Bull, Oracle and the like—and makes them all play nicely (and more cheaply) together.

You don't need to discard and replace those systems with brand-new ones. By identifying the capabilities of existing systems and leveraging them, you maximize the value of your IT investments while minimizing your risk. Also, building services—for example, using simple object access protocol (SOAP) and Web services description language (WSDL)—not only simplifies the internal integration process, it also lets partner agencies share information more easily across agency firewalls.

Another benefit of an SOA is that it can lead to a better dialogue between the IT staffs and justice practitioners by forcing IT workers to think in terms of business—not technical—architectures. If a justice enterprise wants to build a warrant tracking system, for example, the law enforcement and courts folks can hook up with the IT architects and talk about the best way to design it based on business flows and how best to meet the needs of the agencies who handle and serve the warrants. And implementing that design, which often involves large-scale integration, becomes a less gruesome task.

For that dialogue to work, the subject matter experts have to think about the best ways to run their business. What processes do I need to put in place to best accommodate my agency? How can I improve my level of service? By exposing and sharing information



across once-siloed applications, agencies can extract more service performance data in real-time, improving agency intelligence. There's a whole new level of responsiveness companies can exploit through a common architecture.

Finally, the benefits of easier integration and increased agility lead to greater ROI. One example of a justice SOA-based service is an integrated *Search* or *Query* function, in which any agency front-end application can issue a command and, after probing around the legacy systems across the enterprise, come back with a complete picture of a subject's criminal history and contact with the justice enterprise. That's one example of how Justice SOA can improve business and develop ROI—developers design services to be generic enough that they can work with an array of front-facing systems, reducing development time and freeing developers to spend more time on business solutions. In addition, IT workers can easily incorporate new technologies into the SOA, reducing risk and expense while speeding development of new applications.

How Do Web Services Work in an SOA?

First, it's important to note that an SOA does not require Web services; and Web services can be deployed without an SOA. There are purists, however, who believe that building an SOA using Web services is the ideal approach. Gartner Group reflects that definition of a true SOA, but warns that users must implement Web services properly to create a holistic SOA. If done correctly, a Web service is little more than an SOA that uses SOAP and WSDL.

Others have built SOAs without Web services, where no internal or external partners are using them (although most look to employ them later on). Instead, a common construct is to use a proprietary messaging and integration framework, such as IBM's WebSphere MQ to connect legacy systems with their front-end applications.

And still others have built an SOA, based on a “publish-subscribe system” without Web services. Most commonly deploying Java Message Service (JMS) as a messaging layer on top of a Web server and an application server, and using an enterprise service bus to help with integration and data movement. These services are designed like Web services, only without the Web services interfaces.

One of the major benefits of the SOA is that the right data gets sent to the right person or application. For example, when a user logs on using an enterprise approved ID, the system knows who the user is and pushes only the data that the person is authorized to see.

What Are the SOA Challenges?

Trying to manage the complexity of a service[s] configuration can be complex and requires a good SOA governance model. There are many issues across any enterprise



architecture, and a clear governance structure is critical to managing and making decisions over these issues as the system evolves over time.

Security is a fundamental concern. It's always easier to secure a closed system than an open architecture. Enterprises must deal with the lack of security standards for Web services. To overcome some of these security roadblocks, it's still recommended that agencies move slowly when setting up an SOA, focusing first on business processes that don't require a high level of security.

Another issue is network monitoring. As the capability to orchestrate complex Net-centric business processes is created in a service-oriented architecture, complex monitoring and auditing requirements also follow. For instance, when a transaction goes awry on a service-oriented network, which could involve multiple service providers, finding out what went wrong or where the transaction dropped or whether someone put bad information in the network can be a challenge.

Finally, there's the cost issue. Building an SOA isn't cheap; reengineering your existing systems architecture is going to cost some serious money. It also requires significant human capital, including business analysts to lay out the business processes, systems architects to turn processes into specifications, software engineers to develop the new code and project managers to track it all.

Best Practices for Building an SOA

It may sound obvious, but having a blueprint for your SOA is critical. It's very easy for projects, especially large enterprises with disparate operations, to buy new technologies or integrate applications without regard to how they fit into the overall plan. The challenge in building an SOA is to keep people—including both IT and business-side staff—focused on the architecture goals.

IT staff will also need to identify the right level of service to provide. And those services shouldn't have too fine a granularity—that defeats the goal of services, which is to function at a higher, business-process level. Too narrow a focus creates a need for more services, which increases development time. And in the worst case, too many services can flood a network.

