

## Service-Oriented Architecture Primer

Service-oriented architecture (SOA) is an important IT architectural strategy that is changing how software is developed, used and sold. You can use SOA for simple integration, to augment existing applications, to assemble new applications, and even as a springboard to break through the restrictive bounds of traditional application development to business scenario development<sup>1</sup>.

To be successful with SOA, first you need to understand what services and SOA are. From there, you need to adopt solid service-oriented practices in order to build a deep services catalog and implement an extensible SOA environment.

In this page-plus SOA primer, we focus on the beginning, with the basics on services and SOA.

### What is a Service?

Simply stated, a service is a thing that fulfills a purpose. A service is, in essence, a 'worker', employed to achieve a specific end goal for a requestor. The end goal may be small in scope, such as retrieving information, or large in scope, such as executing a business process. Most services are in the middle, completing a function. The scope of a service is referred to as its grain, or level of granularity.

#### *What do we mean by "a service is a thing"?*

The main architectural concept of SOA is that the service is really an abstract resource that has a name, a job, job tasks, contact information and policies regarding security and service levels. The job tasks of the service correspond to physical code assets (objects, components, legacy code, application package API), but only the service knows these specifics. From the requestor's point of view, a service is a (small) black box. To use (request) a service, you send a message – in accordance to the contact information and policies – and then (if appropriate), receive a reply message. The details of the service execution are unimportant to you. You are only concerned that the service does its job and returns understandable results.

#### *What about the service's job?*

The job of a service is limited to a single distinct business concept, function or process. This trait is referred to as the bounds of a service. Finding the correct bounds is a key factor in service definition and building your collection (catalog) of services. A service may call upon other services if it needs assistance in completing its job. This service-to-service relationship is called collaboration.

#### *How do services collaborate?*

Services collaborate through orchestration, business interaction or interception. In **orchestration**, there is always a primary service that directly invokes other services. The primary service knows the sequence of actions, the interfaces, responses and return states from all called services. The sequence of actions may be somewhat improvisational, leveraging rules services to adapt to particular contexts, and business situations.

A **business interaction** is a type of collaboration where some coordination mechanism, outside of the services, knows the sequence of actions, possible states, and paths of interaction between one or more services. The

---

<sup>1</sup> In business scenario development, IT business solutions will be compositions of services, business events, and business processes mirroring the interactions (or flow) of your business.

business interaction is usually long lived and involves (requests/messages) from multiple parties. The coordination mechanism may be a workflow, business process manager or event handler.

**Interception** is a type of collaboration in which an intermediary service is placed between the requestor and the targeted service. The intercepting service does its job, typically an infrastructure job (security, policy, translation, audit) and then forwards the request to the original target. In many interception scenarios, the requesting and providing parties are unaware of the intermediary service.

## Services ≠ Web Service

Services and Web Services share an underlying architectural construct in that each communicate via messages, which use standard notation (XML). However, not all services are implemented using Web Services technology. In addition, some people use Web Services purely for technical integration reasons, forgoing the rules of good service definition. Therefore, the two are similar, but not equal.

## What is SOA?

Architecturally, SOA refers to the loose coupling of a service from its provider. This concept is not limited to functional assets (services). You can apply it to larger IT “services” such as network, platform or even programming services. If a requestor knows what a service offers (job, service levels), and how to use it (contact, security), then it really is not important (to the requestor) how that service works, as long as the results meet expectations. The implementation details of the service are the responsibility of the provider. We refer to this as “service-oriented thinking”. We find this thinking helpful in planning and executing large initiatives (such as architectural adoption), and defining organizational structure and responsibilities.

SOA also refers to the collective environment that allows services to be defined, developed and used by other services, and to be assembled into business solutions by adding process, user interface and business rules. Beyond service development and business solution assembly, the SOA environment provides for service discovery, policy definition and enforcement, quality of service (performance, availability, reliability and load), transaction compensation (undo) and usage metering.

## For More Information:

For more information on SOA or related architecture strategies (event-driven architecture, process-based architecture, enterprise integration) or business scenario development supported by our own business-driven architecture philosophy, please contact us @ [bda@elementallinks.com](mailto:bda@elementallinks.com) or refer to “Our Papers” on [www.elementallinks.com](http://www.elementallinks.com).