



Service Oriented Architecture

A QoS Labs White Paper

September 2007

What is Service Oriented Architecture?

Service Oriented Architecture (SOA) is the result of applying the best Software Engineering practices to the development of complex IT systems. Applications are built using smaller reusable building blocks called services. A good SOA design identifies and defines business logic that can be implemented as a service and can be reused in multiple applications.

Non-SOA Installations

Typical non-SOA enterprise installations have evolved from a set of isolated applications that, in the best cases, are reasonably integrated using either some sort of Enterprise Application Integration (EAI), Enterprise Service Bus (ESB), middle-ware, custom integration development or even manual processes. Since the applications used in these installations were designed in isolation, they duplicate the same basic functions over and over. A significant percentage of the IT development is dedicated to synchronize the status of those duplicated components. For example **Customer** is an entity that appears in most of the applications since it is central to a great number of the business processes. **Customer Synchronization** implementations where changes to the Customer information in one application need to be replicated to the rest of the system, are found practically anywhere an integration project has taken place. It is important to note that all these efforts, in development and operation, do not add value to the business processes. It is just a necessary evil of the technology in place.

SOA Installations

SOA can be used to develop an intranet enterprise system or a multi-organization ecosystem. Although SOA is associated with technology, the success of the implementation depends on how well the team designing the architecture dissects the business processes relevant to the system more than the technology used.

During the analysis and design phases of the system the team must identify common repeatable business tasks that occur in multiple places in the business processes. Those common repeatable tasks or activities become the basic building blocks or **SOA Services** of the architecture. **SOA Applications** are developed then as business logic that implements processes by orchestrating the use of multiple **SOA Services**. Business Processes implemented by SOA Applications can be exposed as reusable SOA Services if appropriate.

From this discussion we can identify some characteristics desirable or required in **SOA Service**:

- A service represents a repeatable, reusable business activity, task or process.
- Each service must define an unambiguous interface. Input data to the service and the possible outcomes must be part of this interface. The possible outcomes must include both success and failure cases. Authentication, access control and policy enforcement should be part of the interface and are often left out.



- A service may be a **composite** service. It is implemented using other services.
- A service implementation should be of no concern to its clients. A service can use any implementation deemed appropriate or available in the system. A service may be automated or require human intervention.
- Loose coupling. Clients of a service should only depend on the interface defined by the service. Implementation and runtime bindings should be transparent to them.
- A service must implement a complete business task or activity.
- A catalog of the services available should be available.

Benefits of SOA

The following is a list of the benefits that can be obtained by using a SOA.

- Business rules applied consistently across the enterprise or system.
- Changes to business requirements will have to be implemented, documented, tested, deployed and monitored in only one place.
- Simplification of the auditing mechanisms needed.
- It is easier, cheaper and more feasible to make changes to the system as business requirements and environments change.
- It is easier to integrate new functionality, business units and business partners.
- It permits the development of new business models. For example per-pay services.
- Development of new applications is performed closer to the business domain and not tied to the underlying technology.

Services Functionality

Experience has shown that the granularity of the services in a successful SOA implementation must be much larger than the typical Object Oriented decomposition into classes and methods. The service must implement a business task or activity, no just expose low level functionality of the components used in the implementation. The orchestration process should use services with a high level of semantic meaning. It should use the business vocabulary not the technical one.

How is SOA Implemented?

As it was mentioned earlier in this paper SOA can be implemented in a multitude of ways, using very diverse tools. You can have a SOA fully implemented by tasks that are executed manually by a human being. In the context of this paper we're interested only in those that are at least partially automated. For those SOA the following guidelines usually apply:

- Do not implement anything until you understand your business processes and you have identified the common reusable SOA Services you want to start building upon.
- Adopt the use of standards where possible. When you talk to different vendors they will present their solution as the 'SOA way' shop around and if possible select the solutions that use established, de-facto or emerging standards in that order.
- Define the services at the right level of abstraction. Use only the business vocabulary. Do not expose any services that use the technical vocabulary of the implementation.
- Create your catalog of services and populate it with their interfaces.



- Hide all implementation details from your interfaces.
- Design reusability and composability into your services.
- Select the technology and tools appropriate for the task. It may be necessary to use different tools to implement different services depending on the history and legacy systems involved. Using SOA doesn't preclude the use of other technologies in the internal implementation of a particular service. For example some services may use a middle-ware solution in their implementation.
- Interoperability. Avoid any platform, product or protocol dependency in your services if at all possible.

No Silver Bullet

Although SOA is considered to be the current state of the art for IT systems and it is arguably a much better alternative to traditional silo development, you need to keep in mind that it is by no means a recipe for success. In some cases it will create issues that do not exist in monolithic applications. Keep an eye for the following:

- The analysis and design is done by the IT folks without participation of the business experts.
- Make sure you have the competencies required for the SOA implementation and methodology in your development team. If not, partner with somebody that does. It is not likely to be trivial.
- Core services that are invoked in multiple applications become critical single points of failure for those applications. You must consider *High Availability* and in some cases even *Geo Redundancy* for those services.
- You should not design your processes using the conventional notion of a *transaction* where you can either *commit* or *unroll*. For SOA application the use of *Compensating Transactions* is much more appropriated.
- Go the extra mile when defining your interfaces. Include the behavior on failure and success as part of the documentation. If the tools you're using do not support this -- and some do not support it yet -- implement it as a convention used as part of your development process.
- Security, Identity, Authorization and Policy Enforcement are key elements of your implementation. Make sure you understand all the implications.

SOA Alphabet Soup

I have mentioned multiple times that SOA may be implemented in very different ways using diverse tools and technologies. The implementation of SOA using Web-Services has won quite a bit of momentum in the industry. the following is a list of acronyms you are likely to find while diving into the field. Again it is not by far a complete list of technologies.

- XML. Extensible Markup Language. It is a general purpose markup language very popular in internet related applications.
- WSDL. Web Services Description Language. It is an XML based language to describe the interface to a Web Service.
- SOAP. Simple Object Access Protocol or lately Service Oriented Architecture Protocol. Is a protocol for exchanging XML messages or request/responses over computer networks.
- REST. Representational State Transfer. Architecture for defining and addressing network resources. Commands to the resources are defined in simple terms.



- UDDI. Universal Description, Discovery and Integration. It is a registry for Web Services. It uses WSDL and SOAP in its implementation.
- BPEL. Business Process Execution Language. Is a business process modeling (BPM) language. One of the intended applications of BPEL is to orchestrate services exposed using Web Services defined in WSDL.
- WS-Choreography. Web Service Choreography. BPM language for collaboration of cooperating Web Services.
- WS-CDL. Web Services Choreography Description Language. XML based language that describes peer-to-peer collaborations of parties by defining, from a global viewpoint, their common and complementary observable behavior; where ordered message exchanges result in accomplishing a common business goal.
- WS-* Diverse Web Services specifications. Various Web Services specifications that may be applied in an SOA implementation. Among them WS-Eventing, WS-Addressing, WS-Enumeration, WS-Transfer, WS-I Basic Profile, WS-BPEL, WS-Policy, WS-Discovery, WS-Security, WS-Trust, WS-Federation, WS-Transaction, etc.

Summary

SOA is here to stay and represents the evolution of the best practices in IT. SOA is a way to approaching application design and it can be implemented using a variety of technologies. Although it is a much better alternative to the traditional IT systems development, implementing SOA in your environment is not a walk in the park. Invest the time and resources to build the required competencies for the endeavor. You should also strongly consider partnering with experts in the field.



System Oriented Architecture
September 2007

Author: Hermán de J. Camarena R.

QoS Labs
621 NW 53rd Street, Suite 260
Boca Raton, FL 33487
U.S.A.
+1(561)988-8020
www.qoslabs.com

Copyright © 2007, QoS Labs. All rights reserved.

The contents of this document are subject to change without notice.

This document is not warranted to be error-free. We disclaim any liability and accept no contractual obligations resulting from this document.

This document or parts of it may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without QoS Labs' prior written permission.
