White paper
Why should I care about SOA?
Introduction

As every Oracle® professional knows, Oracle’s Technology Network (http://www.oracle.com/technetwork) is an invaluable resource for working with Oracle technologies. Because of Oracle’s aggressive acquisition strategy over the past couple of years, the product set offered by Oracle has grown to include various development technologies (Java®), business intelligence tools (such as Oracle Business Intelligence Enterprise Edition (OBIEE) and Essbase®), various enterprise programs (such as PeopleSoft® and JD Edwards®) and other enterprise tools. This dramatic increase in Oracle’s portfolio makes Oracle the premier company for end-to-end enterprise software solutions.

The above-mentioned resource is great for figuring out the “how”:

How do I set up SSL on the application server?

How do I write a materialized view?

How do I create a composite application in JDeveloper?

What can be a real challenge for Oracle professionals are the “Why” questions: Why should we go down this path? Why should we invest resources in these technologies? Fujitsu can bring clarity to the many questions facing organizations looking to implement Oracle SOA-based solutions.

What is SOA?

SOA means a lot of different things to different people, but it comes down to two basic principles:

1. **Philosophy:** SOA requires everyone involved in the solution to think differently about how complex systems are put together. In traditional development, a methodology regarding the development of business requirements, coding standards, testing standards, implementation of the final application and acceptance by the end users is adopted by the development team. Once the methodology is decided upon, the tasks of requirements gathering, application design and coding are usually the first three activities to begin. In a true SOA-based application, however, the application design phase takes on much more significance. SOA encourages developers to break parts of an application into small, reusable pieces of code called business services. These business services typically perform a very specific business function and are usually implemented as a web service (discussed below). These web services can then be assembled into applications (referred to as composite applications) or reports.

2. **Technology:** The implementation of SOA isn’t a “thing”. There’s no button in Oracle JDeveloper that says “Generate to SOA”. Developing and implementing an SOA-based application means adhering to a set of standards, and when it comes to SOA, there are a lot of standards. One of the challenges for developers and architects is not only being aware of the different standards, but knowing when to choose between the different types of implementations available to you. For instance, let’s take the simple example of a piece of code that validates a customer’s information. Where should that code live? Here are some of the places it could live in a SOA-based application:

- In the browser code (typically in a .jsp)
- In the Java application (deployed to the middleware server – for Oracle, this would be the WebLogic Server)
- In a Web Service
- In an Entity Java Bean (EJB)
- In the Enterprise Service Bus
- In a BPEL process
- In a Business Rule
- In a stored procedure in the database
Why should I care about SOA?

There are standards when it comes to designing and implementing a web service, a BPEL process, an EJB, etc. Knowing about all of these standards and making sure your code adheres to them is an absolute requirement of SOA-based application development. The whole premise of SOA is based upon adhering to these standards. Any violation will result in an unstable system that will be impossible to maintain and enhance. Fujitsu can partner with your developers and architects to produce a system that is responsive, secure and extensible to serve the needs of your users now and into the future.

Why should I care about SOA?

There are two aspects of this answer. The first regards your status as an Oracle professional. Oracle has embraced Service-Oriented Architecture across all of its products. In a 2008 keynote at Oracle OpenWorld, Larry Ellison said that “all Oracle products will be SOA-enabled”. As an example of Oracle’s commitment, they purchased the leading application server on the market (BEA’s WebLogic®) and made it the middleware standard for all Oracle products moving forward. They have also undertaken one of the largest development projects in history (Fusion) that uses SOA extensively to integrate the enterprise software systems that Oracle has purchased recently (JD Edwards, PeopleSoft, et al.). It is safe to say that Oracle has invested a tremendous amount of money and resources embracing SOA technologies. Does that mean it will always be the case? Of course not, but if you deal with Oracle technologies, chances are that you will be required to have at least a fundamental understanding of SOA if you would like to stay relevant in your job.

The second aspect is not Oracle-specific and deals with the complexity of modern informational systems as a whole. To fully understand some of the drivers behind SOA, we need to take a step back and look at something called architecture.

A Brief History of Computer Architecture

Architecture is a fancy word for how computer systems (and how end users interact with those systems) are put together. In the infancy of business computing there were few choices as to how a system was “put together.” Like everything else, this is a double-edged sword: on one hand there was virtually no flexibility, but on the other hand, it was (relatively) simple to design and implement. In modern systems, there are virtually an unlimited number of permutations for constructing computer systems giving businesses incredible flexibility. With this flexibility, however, comes great risks including: creating architectures that can’t grow with the business, security issues at all levels of the architecture, finding and keeping knowledgeable workers to support and maintain these complex structures, aligning business goals with technology, getting locked in to a vendor’s technology, keeping abreast of technological changes and offerings – the list goes on and on. Below is a list of the major business computing architecture eras and how they have evolved over the past 60 years.

<table>
<thead>
<tr>
<th>Era</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>1950’s – 1980’s: The Dumb Terminal</td>
<td>Simple</td>
<td>Not Scalable</td>
</tr>
<tr>
<td>1980’s – 2000’s: Client/Server Computing</td>
<td>Distributed Computing</td>
<td>Difficult administration of applications</td>
</tr>
<tr>
<td>2000’s: 3- and n-tier computing</td>
<td>Distributed computing and easier administration</td>
<td>Complexity of applications / lack of standards</td>
</tr>
</tbody>
</table>
So we've solved all of the major architectural computing issues, right? We have flexibility to construct systems in a myriad of different ways with components from different vendors. We have the ability to deploy applications to a single location. We're not tied to a client operating system, or even a type of client (PC application, PC browser-based, mobile computing, etc.). We can scale systems with relative ease through both clustering and architecture design. With all of these enhancements, why is SOA making inroads in many organizations today? Even with all of these enhancements, there are three main challenges we still need to consider: the size of most modern applications, integration and the alignment of business goals with technology implementations.

Consider the sheer size of most enterprise applications. Large enterprise applications like Oracle's e-Business Suite are made up of thousands of database tables, thousands of web pages and functionality that spans virtually every aspect of every employee within an organization. Tools like source code control and project management greatly assist in the development and implementation of new development, but the sheer size of many development projects outstrips tool and human capacity for managing these projects successfully. What if organizations were to adopt a SOA-centric approach for their development? If you remember earlier, one of the main philosophies of SOA development is to define specific business functions and implement them as web services. Once the web service is deployed, its information can be "published." Developers can then easily reuse functionality already defined (and presumably tested) to construct new applications and functionality easily. These web services can also be monitored and have security rules associated with them to provide additional functionality and governance to an organization.

The next challenge deals with integration. Organizations typically look for best-of-breed applications to support their business. This usually results in disparate systems that don't natively speak to each other. By implementing an SOA-based philosophy towards application development, composite applications that use web services tied to disparate systems can be developed easily. Since a web service is language agnostic, the composite application that brings together information from these disparate systems has no concerns as to how the web service was implemented. This point goes back to the earlier rule that standards must be followed in order to ensure a successful SOA implementation. If the standards are followed, composite applications can access these disparate systems without any fear of compatibility issues. Also, consider the need for most organizations to interact with outside entities (suppliers, vendor, government agencies, etc.). If the outside agency is willing to provide a web service, your organization could interact with those external entities without having to worry about how the external entity's systems are coded – language, operating system, underlying enterprise software version would all be moot. The published web service would detail how to interact with the underlying system and applications could be developed against that web service definition.

The third challenge deals with business and technology alignment. Since the beginning of business computing, the disconnect between business types and technology types has been a challenge many organizations have struggled with. The simple translation of a business rule into a piece of code that accurately reflects and enforces that business rule has caused more problems for IT departments than virtually all other challenges combined. Part of the SOA standard implements technologies that allow business analysts to define business rules that affect how the underlying system works. By allowing business analysts the ability to influence how a system functions, the necessity of translating business rules into executable code is diminished (and in some cases, removed altogether).
Why do Web Services exist?

As we’ve seen, one of the basic SOA tenets is to follow the basic philosophy of every “Intro to Programming” college course, namely this: break big problems into small, manageable chunks. Most programming languages have the concept of a procedure, function or library that allows you to develop a piece of code that can be used in multiple places without having to duplicate functionality. Web Services takes this concept one step further by defining rules regarding the following:

- How the “chunk” of code is called
- The types of data expected as input and the types of data provided as output
- A protocol for being able to call this “chunk” of code over the Internet (technically over a network protocol like TCP/IP)
- A standard way of publishing information about this “chunk” of code so developers know how to call it without knowing anything about how the chunk of code was written
- Security rules for who can call and run the “chunk” of code

So a web service exposes a small piece of a large system over the internet. We can then create a type of application called a composite application that uses these web services to perform some sort of business function. Consider an application where you keep information about vendors. One of the things you might want to keep track of is a vendor’s credit rating. You might write the “check-credit-rating” code inside the application OR you might turn the “check-credit-rating” code into a web service. If you do the latter, the next time you need to write an application that checks a vendor’s credit rating, you know you already have the code for it. Not only that, but it’s been tested and in use. You could then monitor that web service – it could be to another part of the business’s advantage to know how many times credit checks were run on a vendor, or over a period of time, what percentage of credit checks were declined (or below a certain value), etc.

You could also enhance the web service to not only check credit ratings for vendors, but for individual customers. The web service would be the one place for all types of credit checks within an organization – this would make it much easier to maintain, enhance and monitor, as opposed to doing these types of things across disparate systems with disparate rules with disparate terminology programmed in disparate programming languages. And since it’s a web service, you’re not locked into a technology or programming language – if the decision is made to switch from Java to Microsoft .NET, your existing Web Services are still valid and can be called from any language that supports web services (of which all major modern programming languages do).

Why does the Service Bus exist?

The Service Bus is a special application deployed by most SOA-based systems. The Oracle Service Bus (OSB) performs a lot of functions, but let’s start with what functionality the OSB provides to Web Services. When it comes to Web Services, you can think of the OSB as a proxy server for Web Services. But then the question becomes, why do Web Services need a proxy server?

A proxy server sits between a client and the resource a client wants. A proxy server can be used to filter requests or provide some other sort of transformational logic between the client and the resource the client is requesting. The OSB can act as this intermediary between a request for a Web Service and the Web Service itself. Let’s say an external company publishes a Web Service and you write an application based on that Web Service. The external company then gets bought by another company and they provide their own Web Service that’s different from the original company’s Web Service. What happens to your composite application? It breaks, since the new Web Service has a different set of parameters and calling method. One of the functions the OSB can serve is, as a proxy that “maps” your call to the original Web Service to the new Web Service. The underlying Web Service returns information to the OSB, which then maps the data back to your calling application.

That’s not the only thing the service bus can do. The OSB also serves as the communications nerve center among all applications that pass messages back and forth between applications and requests. Rules can be set up that support the following types of messages:

- Point-to-point
- Point-to-point request/response
- Broadcast
- Broadcast request/response
- Publish/subscribe
- Store and forward
The OSB provides a way to manage and version Web Services. It also provides adapters to other systems that allow messages to be sent and received (and possibly transformed) from and to other systems easily. OSB also provides advanced manipulation of message via a piece of functionality called the message broker. It mediates communication amongst applications, minimizing the mutual awareness that applications should have of each other in order to be able to exchange messages, effectively implementing decoupling. The following are examples of actions that might be taken in the broker:

- Route messages to one or more of many destinations
- Transform messages
- Perform message aggregation, decomposing messages into multiple messages and sending them to their destination, then recomposing the responses into one message to return to the user
- Invoke Web Services to retrieve data
- Respond to events or errors
- Provide content and topic-based message routing using the publish/subscribe model

The OSB can also provide authentication, authorization, privacy, integrity and auditing services for messages.

While all service buses are slightly different in terms of the functionality they provide, almost all provide the following additional services:

<table>
<thead>
<tr>
<th>Invocation</th>
<th>Support for synchronous/asynchronous transport protocols, service mapping (locating &amp; binding)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing Mediation</td>
<td>Addressability, static/deterministic routing, content/rules/policy-based routing adapters, protocol transformation, service mapping</td>
</tr>
<tr>
<td>Process choreography</td>
<td>Implementation of complex business processes</td>
</tr>
<tr>
<td>Service orchestration</td>
<td>Coordination of multiple implementation services exposed as a single, aggregate service</td>
</tr>
<tr>
<td>Complex event processing</td>
<td>Event-interpretation, correlation, pattern-matching</td>
</tr>
<tr>
<td>QoS Management</td>
<td>Security (encryption and signing), reliable delivery, transaction management monitoring, audit, logging, metering, admin console, BAM</td>
</tr>
</tbody>
</table>

**Why is there a Rules Engine?**

Consider what a simple enhancement would entail for a development environment in a typical IT organization using an enterprise software package like Oracle e-Business Suite:

- User identifies need(s)
- Business analyst outlines change
- Developer must locate change and implement
- Unit test for validity
- Integration test
- System Test
- Regression test
- Acceptance test
- Deployment

Now consider how long those steps would take in a complex, distributed environment governed by federal and industry standards and laws. On top of all that, add in the complexity of translating business rules into code, the human interaction between business-types and technical-types and the different interpretations of terminology and you’re left with a situation where the chances of success grow smaller and smaller.
Why should I care about SOA?

What if, by contrast, there was a way for business analysts to define business rules directly in the system? Not only that, but what if they could define, alter and test these rules on the fly? Steps 4 through 9 in the list above could be eliminated. Would that serve as something that would appeal to most businesses?

The Business Rules Engine allows you to do just that. Applications can be set up to reference rules, instead of hard-coded business parameters. These rules can then be maintained by business analysts, who can make changes to the rules without affecting the application whatsoever. The next time business logic dictates a particular decision inside an application, the newly altered business rule will automatically take effect, shortening the time between identification of a change and its implementation while all the while leaving IT out of the loop.

Why is everyone talking about “the cloud” and how does SOA fit in?

How confident are you in your estimating abilities? Most of us, even with years of experience under our belts, are lousy at estimating, and with systems getting more and more complex, the estimating abilities of all but the most talented of professionals is progressively getting worse. Of all of the benefits of the cloud (24/7 availability, redundancy, reduced costs), the ability to “spin up” new hardware at (virtually) a moment’s notice, may be the most important, particularly for companies enjoying growth in these lean times. By taking the hardware procurement process (and its increasingly complex system of hardware contracts and endless legalese) out of the equation, developers and IT departments can focus on getting things done, instead of supporting the increasingly time-consuming (and soul-deadening) tasks related to hardware acquisition, administration and maintenance.

Since one of the main tenants of good SOA design is to break code into small, reusable, distributed “chunks”, it dovetails into cloud-computing principles very nicely. Web services can be moved amongst systems very easily and since they (along with all SOA components) are dependent on adherence to standards, they can be deployed to different application servers on different hardware running different operating systems easily (in theory, of course – in real life, there are still incompatibilities). This flexibility, along with the ability to “spin up” or “spin down” hardware at a moment’s notice gives organizations incredible flexibility to architect their environments.

Conclusion

Service-Oriented Architecture, while complex, provides organizations with the most flexible, business-focused architecture of any technology today. Coupled with Oracle’s embrace of SOA in its Fusion initiative, all Oracle professionals would be well served by a fundamental understanding of SOA and its basic concepts. Effectively communicating the benefits of SOA and its design fundamentals will be a pre-requisite for all Oracle professionals hoping to remain relevant in the future.
ABOUT FUJITSU AMERICA
Fujitsu America, Inc. is a leading ICT solutions provider for organizations in the U.S., Canada and the Caribbean. Fujitsu enables clients to meet their business objectives through integrated offerings including consulting, systems integration, managed services and outsourcing for enterprise applications, data center and field services operations, based on server, software, storage and mobile technologies. Fujitsu provides industry-oriented solutions for manufacturing, retail, healthcare, government, education, financial services and communications sectors.

For more information, please visit: http://solutions.us.fujitsu.com/