Chapter 1

SOA and Web Services Introduction

Objectives:

- Define Service Oriented Architecture (SOA).
- Take a look at the advantages that SOA based systems provide.
- Define Web services.
- Introduce the Web services technologies.
- Introduce the Web service standards organizations.
Chapter Overview

Service Oriented Architecture (SOA) fundamentally changes the way IT teams design their applications and establishes an environment of reuse. What is SOA and how does it help organizations think about and then implement software systems?

Web services are often mentioned in the same sentence with SOA. They are related, but not the same. What are Web Services and how do they compare to SOA? In this chapter, Web service technology is explained. You will also explore XML Schema, SOAP, WSDL and UDDI, which are the key specifications that define Web services.
Service Oriented Architecture (SOA)

- **What is SOA?**
  - You have probably heard the acronym SOA before, but what does it mean?
  - SOA is an architectural style -- otherwise known as an architectural pattern.

- **Patterns are “a common solution to a common problem in a given context.”**
  - Patterns allow us to store and share knowledge of proven, acceptable solutions to problems.
  - Patterns can address a wide variety of problems.
  - There are all sorts of patterns, both inside and outside of software engineering.

- **In software, patterns are generally organized into three groups or types:**
  - Architecture patterns (a.k.a. architectural styles)
  - Design patterns
  - Language idioms
  - (Note the terms architecture, design and idioms are generally accepted, but different names have been applied to the same general pattern ontology.)

- **The difference between these types is their levels of abstraction and detail.**

- **Architecture patterns are broad in scope.**
  - They are higher-level patterns describing the high-level strategies of a system.
  - They address global or system-wide properties of the system.
  - They often impact the overall system in a fundamental way.
  - Think of architecture patterns as “sets of principles that shape an application.”
  - Some examples of architecture patterns (beyond SOA) are layers, tiered, pipes/filters and even model-view-controller.

---

2 [Pattern-Oriented Software Architecture](https://books.google.com/books?id=buschmann), Buschmann et al. (John Wiley & Sons, 1996)
• **Design patterns are medium in scope.**
  
  • Design patterns specify the structure and behavior of a group of classes and other elements. In other words, design patterns are a solution to a part of a system.
  
  • Design patterns “reside in the domain of modules and interconnections.”
  
  • Design patterns do not often impact the overall system in any kind of fundamental way.
  
  • “They do not influence overall system structure, but instead define micro-architectures of subsystems and components.”
  
• **Design patterns were made famous by the book *Design Patterns: Elements of Reusable Object-Oriented Software.*
  
  • Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides authored the book.
  
  • They are known today as the “Gang of Four” and their book is often called the “Gang of Four book.”
  
  • Example design patterns (from the book) include the Singleton, Factory Method, and Adapter.

• **Idioms are small in scope.**
  
  • Idioms are low-level patterns that are usually specific to a programming language.
  
  • Idioms capture programming experience – not really design experience.
  
  • Idioms help to prevent common problems or pitfalls with a programming language or help teach a unique feature of a language.
  
  • Idioms are often loosely translated to best or standard practices.
  
  • A warning not to use too much of the String concatenation operator (+) in Java due to its inefficiencies and excessive garbage generation is an example of a Java idiom.

---

4 [en.wikipedia.org/wiki/Software_design_pattern](en.wikipedia.org/wiki/Software_design_pattern)

5 [www.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-02-03/aswe14-essay.pdf](www.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-02-03/aswe14-essay.pdf)
SOA – the architectural pattern

- So, if SOA is an architecture pattern, and patterns are common solutions to common problems, what are the problem and the solution here?
  - In many large companies, there are a lot of similar software applications that do the same (or very similar) business processing.
  - Each time a department or branch wants something, it is typical that a new software application is built to address that need.
  - Sometimes the need to build the new software is deemed necessary because of technical advancements. “We need to move this from the mainframe to the Web.”
  - Other times, the department or branch members believe their need is unique enough to justify a new version of the software.
  - Eventually, the organization has lots of software that provides for the same services, sometimes with slight variations and often built on different platforms.
  - Mergers and acquisitions complicate matters, as these organizations come with their own duplicate software applications that provide the same or similar services.
• **This set of systems becomes very difficult and expensive to maintain, and offers little in the way of reuse to help dampen the cost of any new system.**

  • What’s more, a change in business policy or need to affect new business strategy becomes very hard and expensive to unilaterally implement.

  • It is first difficult to find all the software that applies to the business change. Once found, making all the changes and testing them is expensive and time consuming.

  • Software applications built in silo fashion to meet business needs end up inhibiting businesses that aspire to be more dynamic, innovative, and agile.

• **SOA’s solution to this problem is to address software applications not as silos, but as a collection of business services.**

  • A *business service* (or simply *service*) is defined as a distinct, self-contained (or sealed) unit of automation logic that provides a specific business process.

  • Each business is different, but a business service might be “Create Invoice” or “Make Reservation” versus something like the General Ledger app or Travel Web site.

  • In service-oriented architectures, the business service becomes the one and only place for any particular business function, policy, or rule.

  • The single business service gets used everywhere in the company that needs the function offered by the service.

  • When a business change is required, there is only one place to make that change in code.

  • Software applications are formed by connecting various reusable business services.
Intuitively, you know this makes sense. Think of the way your company works.

- Does each department or sub-organization within your company have its own travel department, accounting group, legal advisors, or human resources personnel?
- More than likely, these “services” have been consolidated under the corporate structure.

There is one travel department, for example, which all departments seek for travel assistance.

This reduces overhead costs, improves efficiency and ensures corporate travel policy is adhered to universally.

Sorry – no first class accommodations on your next business trip despite what your manager may have promised.
How is SOA accomplished? The SOA Principles

- Conceptually, SOA is quite simple. SOA is based on software serving one of three roles:
  - Provider of a service – a business service/a unit of automation logic.
  - Requestor of service(s) – also known as a consumer or client.
  - Registry (or broker) of service(s) – a manager of a list of business services for requestors.

- On this basis, one can see how SOA can be used to integrate applications, or it can be used to build new applications comprised of several services.
  - In fact, SOA blurs the line between application integration and application development.
  - What defines an application? What defines integration? Would it even matter if some of the services were provided by third parties running outside the organization?
• **The devil is in the details. While architecting software around core business services sounds good, there are challenges.**
  - How do you know what business services are available?
  - How do you get what you need from the service even if it is available? In other words, how do you “talk to” the service?
  - What if the service is written in .NET and I am building my department’s Web site in Java?
  - What if there is a unique department need that requires just a slightly different implementation from the corporate business service?
  - What happens when the business service needs to be upgraded and/or it moves to a new location?

• **All good questions. SOA isn't just about building software in terms of business services.**
  - SOA is also about building business services that conform to a set of specific characteristics, or as some say, adhere to certain SOA principles.
  - Without adherence to these principles, there may be a collection of units of automation logic without the ability to assemble them into effective applications.
• **Thomas Erl, one of the preeminent authors and speakers on the topic of SOA** has defined eight service-orientation principles.

  - Contract based. Services must express their purpose and capabilities via a service contract or a service description document.
  - Loosely coupled. There are several facets to the loosely coupled design principal.
    - Consumers should be minimally impacted by changes to the service or its environment.
    - Services should minimize dependencies on other services.
    - Services should minimize dependencies on their environment (hardware, location, etc.).
  - Abstract. Except for those details provided through the service contract, service implementation details should be hidden from consumers.
  - Reusable. The division of business logic in services should promote and maximize reuse throughout the enterprise.
  - Autonomous. Services control their internal logic, environment and resources.
  - Stateless. Services minimize managing state (storing any internal data) on the part of the consumer. This minimizes service resource consumption and scales better.
  - Easily discovered. Services must communicate meta data that allows consumers to be able to easily find relevant services.
  - Composable. Higher level services should be able to be created from orchestrating the work of several lower level services.

• **Additional principals have been suggested by other SOA experts. A few additional principles are listed here.**

  - Location transparency. A consumer should be able to use a service without knowing where the service is running.
  - Extensible. A service is more apt to be reused if it can be extended without affecting existing consumers.
  - Normalized. Services are decomposed/consolidated to minimize logic redundancy.
  - Optimized. Services are structured in some cases to provide performance or access optimization.

---

6 serviceorientation.com/index.php/serviceorientation/index
Benefits of SOA

- **The benefits of SOA should be obvious.**
  - Software componentry is more aligned to the business and is more able to meet the fast, dynamic changes that occur in business.
  - Business practices and policies become codified in business services, leading to better adherence to business directions and goals.
  - There is better reuse of software. In fact, most software is, strictly speaking, the reuse and reassembly of services into new applications.
  - There is a lack of redundancy which leads to better scalability, consistency and maintainability of software.
  - SOA can extend the life of legacy systems, as consumers of a service should be indifferent to the underlying implementation of the service.
  - Architectural decisions can be streamlined. Selection of vendor supporting products, platforms, etc. can be made based on how they support services.
  - SOA supports best of breed solutions in that there is only room for one of each business service (and supporting platforms). You have to be the best to survive.

- **For those more-senior IT professionals, you may tend to think there is “nothing new here.”**
  - Indeed, the concept of a single module/component for any piece of business knowledge that is reusable across the application portfolio is not new.
  - Subroutine/component libraries, object orientation and various forms of distributed computing have all tried (to some extent) to address the same issues.
  - Is there anything different in SOA? Aren’t we doing this already?

- **Yes, in that SOA is based on building reusable components (a.k.a. services).**
• **No, in that the level of granularity shifts from other technologies.**
  
  • SOA is not about reusable low-level components (or objects or subroutines) that only developers know/worry about.
  
  • SOA is about high-level business services that even the business managers should care about.
  
  • As already indicated, SOA is more business-centric. SOA business services are designed (in part) by business experts and implemented by engineers.
  
  • In fact, where SOA fails, there is often not enough participation from the highest levels of business.
  
  • The passage below underscores the granularity and shift of thinking that must occur to make SOA an effective strategy in an organization.

If you’re business management, don’t turn SOA over to the IT organization and wash your hands of it. If you are IT management, partner with business management. For SOA to be effective, it must be done from the top down. In other words, if you really want your SOA plan to succeed, business management and IT must work together.  

---

Web Services and SOA

- **Web services and SOA are thought of synonymously.**
  - They are related, but not the same.
  - Web services are a form of SOA implementation.
  - SOA doesn't require Web services.
  - Web services are a set of implementation standards designed for service-oriented architectures.
  - The SOA principles have made Web service technology a good implementation strategy, but it is not the only implementation strategy.
  - REST (Representation State Transfer) based services are another popular implementation strategy (at least as defined by many industry people).

- **SOA assumes no particular technology or standard. It is just a style with a prescribed set of architectural principles.**
  - Web services put SOA into practice.
  - Web services are SOA with discipline and structure.
Further, use of Web service technology also does not mean you are doing SOA.

- Organizations use Web service technology to create a type of data exchange (internally or externally) without creating a collection of reusable business services.

For example, a real estate organization may make its property listing data available to third-party Web sites/applications via a Web service.

- Many industry experts would not consider this single Web service a form of SOA.
- However, there are no SOA police and definitions have been subject to interpretation.

NOTE: Be careful when using the term “Web services.”

- During this class, the term Web services will be used to refer to SOAP-based Web services that follow the WS* set of specifications.
- You learn more about WS* in just a bit.
- However, in another context, Web service may mean RESTful Web services or more broadly, other forms of SOA implementation.
Web Services

- **What are Web services? Why are they popular? Why should you care?**

  - **A Web service is a unit of automation logic that receives messages and responds to those messages.**
    - The messages are typically sent through the Web using HTTP (the transport protocol).
    - The messages are typically formatted in XML.
    - The messages typically follow the SOAP standard (the message protocol).
    - The services are described using a Web Service Definition Language (WSDL - pronounced wiz-dull) document.
    - Potential consumers of a service can find Web services using a registry following the Universal Description Discovery Integration (UDDI) standard.

- **Web services are a popular form of SOA implementation for many reasons.**

  - **It is an XML base technology and XML is agnostic.**
    - XML messages can be created and consumed by any language on any platform.
    - WSDL documents describing a service in XML can be used to create code in any language on any platform.
    - This allows a team to implement a Web service, or Web service consumer, using any comfortable platform/language (Java EE, .NET, COBOL).

  - **Its specifications are not-proprietary.**
    - SOAP, WSDL, UDDI and related specifications have created a non-proprietary way to build distributed, reusable, business services.
    - Standardization allows you to integrate different technologies and platforms like Java EE, .NET and the mainframe.

  - **While not required, many Web service implementations allow organizations to use existing low-cost Internet infrastructure.**
• Of course, there is no free lunch in software engineering.
  • All this “good stuff” comes at a cost.
  • Performance and complexity are the cost.
  • Studies have been done that indicate RMI, for example, is more than 10 times faster than Web services in the exchange of data.\(^8\)
  • While it is the intent of this class to help simplify and make clear Web services technologies, arguments that Web services can be difficult have some merit.
  • In fact, RESTful Web services and similar approaches have emerged to address some of the complexity.

\(^8\)www.semgrid.net/Citation-Before-2006.1/+++JSS-2006-Service.pdf
Web Service Standards

- To understand Web services, begin by investigating the standards that are core to almost all Web service implementations.

<table>
<thead>
<tr>
<th>Web Service Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
</tr>
<tr>
<td>XML Schema</td>
</tr>
<tr>
<td>SOAP</td>
</tr>
<tr>
<td>WSDL</td>
</tr>
<tr>
<td>UDDI</td>
</tr>
</tbody>
</table>

- As mentioned, XML is a fundamental technology in Web services.
  - It is used in the messages between consumers and services.
  - It is also used for describing a service and the data exchanged in a service.
  - When looking to add security or addressing, you add new XML tags to messages.
  - When attempting to locate or publish service information, you use XML to transfer that information.
  - If you dislike XML, get out of SOAP based Web services – NOW!
• **XML Schema is a language for describing the structure and constraining the contents of XML documents.**
  
  - Schema defines the legal building blocks of an XML document.
  - Defining the elements and attributes that can appear in a document.
  - Defining the relationship, order and number of the various XML elements.
  - Defining data types for elements and attributes.
  - In short, XML Schema defines the data that is in the XML documents.

• **SOAP is a standard that defines how to send messages using XML.**
  
  - Messages can be informational messaging (service – here is my status and/or some data you need to know).
  - Messages can also represent requests for some service (service – I need something accomplished and/or need some information).
  - SOAP is an XML-based specification.

• **WSDL expresses the purpose and capabilities of the Web service. It defines the service contract of the Web service.**
  
  - WSDL defines how to communicate with the Web service. It defines the operations a Web service can provide to its consumers.
  - WSDL defines what data is exchanged between the Web service and its consumers.
  - WSDL enables loose coupling between services and consumers.
  - WSDL and Schema provide the impetus for tool developers to generate code that consumers can use to communicate with a service.
  - WSDL is also an XML-based specification.

• **The registry and lookup of services is also standardized in Web services.**
  
  - UDDI (Universal Description Discovery and Integration) is a standard aimed at producing a way to find Web services.
  - UDDI uses (unsurprisingly) XML and SOAP to describe the services available.
  - UDDI defines a registry service for publishing business entity information and the services they offer.
Standard Managers

- **The key to the success of Web services is standards.**
  - Standards provide the interoperability in many programming languages and on multiple platforms.
  - There are several groups at work to develop, maintain and improve Web service standards.
  - As you look at the technologies and standards that guide Web services, you also look at the group that manages the standard.
- **In general, many of the Web service standards are managed by a couple of central organizations.**
- **The World Wide Web Consortium (W3C) is the predominate manager of the key specifications, including XML, XML Schema, SOAP and WSDL.**
  - The W3C is the same group that manages other Web technologies like HTML.
  - In fact, the W3C is headed by Tim Berners-Lee.
  - Berners-Lee is the original creator of URL, HTTP and HTML (the principal technologies that form the basis of the Web).
- **A few of the specifications, like UDDI, are managed by the Organization for the Advancement of Structured Information Standards (OASIS).**
  - OASIS is an international not-for-profit consortium of companies that want to improve e-commerce.
  - In particular, their purpose is to drive the development, convergence, and adoption of e-business standards.
  - In addition to Web service standards like UDDI, OASIS has helped develop OpenDocument format and PKI standards.
- **Developing communications and integration between disparate systems in many languages and on many platforms has been a huge undertaking.**
• An industry consortium, the Web Services Interoperability Organization (WS-I), formed to aid in the development of interoperable Web services.
  - The member list of WS-I reads like a who’s who of technologies: Microsoft, IBM, Oracle, VeriSign, etc.
  - The WS-I is now part of OASIS, and is called OASIS Web Services Interoperability.
  - The WS-I has created several specifications to codify how to create web services, including the popular “Basic Profile.”

• The Basic Profile is a set of implementation guidelines on how Web services technology like Schema, SOAP, WSDL, and UDDI should be used.
  - Specifically, the Basic Profile is a set of recommendations to developers on how to develop *interoperable* Web services.
  - They work with the existing specifications, seeking to clear up inconsistencies and unclear requirements.
  - They do their work without requiring changes to the fundamental specifications and thereby avoid the slow moving W3C spec rewriting.
  - To better assist organizations to live up to the recommendations in the Basic Profile, WS-I also provides a set of analysis and testing tools.
  - These tools are free, and can help determine whether your Web service documents conform to the specifications.
• Most organizations abide by the specifications defined by W3C, OASIS and the WS-I's Basic Profile.
  
  • In addition, there are extensions to these standards that address Web service security, addressing, messaging, etc.
  
  • Many of the specification names in these areas usually start with “WS-“.
  
  • Therefore, collectively all the Web service specifications are typically referred to as the WS* or WS-splat specifications.
  
  • Note, however, this is not an official name and there is no consistent recognition of what specifications fall in/out of this moniker.
Java and Web Services

- **Java is just another platform and language that can participate in providing Web services and in creating Web service consumers.**

- **Java provides tools and technology to work with all the Web services standards; namely XML, SOAP, WSDL, UDDI, etc.**
  - There is an array of APIs to work with these Web service raw ingredients.
  - JAX-WS, JAXB, SAAJ, and JAXR to name a few.
  - Some of the Web Service or XML tools, technologies and API you may use directly when building a Web service or its consumer.
  - Other APIs are utilized primarily by vendors that help provide the tools and infrastructure in support of Web service development and runtime environments.

- **While it is important to understand Web service standards, Java provides Web service technologies that allow you to work at a higher level.**
  - That is, Java provides technology to help abstract away many of the details of Web services. This often includes hiding the distributed nature of Web services.
  - These APIs, tools, etc. even remove you from having to constantly work directly with/in the Web service XML technologies like SOAP, WSDL, etc.
  - Your Java-based services or consumers communicate as if making simple method calls to a local component versus making a distributed call with XML.

- **The first half of this class is dedicated to teaching you XML-based technologies that are instrumental to Web services.**
  - A fundamental understanding and appreciation of the XML technologies is critical to understanding Web services and interoperability.
  - In the second half of the class, you learn how Java provides a framework (API and tools) to build Java Web services and clients or consumers of Web services.
  - While the framework helps abstract away many of the XML details, understanding the XML technologies is still critical to debugging/tuning more complex services.
General Resources

- **Java SOA Cookbook**, Eben Hewitt (O'Reilly 2009)
- W3C XML Web site: www.w3c.org
- OASIS Web Site: www.oasis-open.org
- WS-I Web Site: www.ws-i.org
- SOA Web Sites
  - www.servicesoriented.com
  - www.zapthink.com
  - www.serviceorientation.org
  - www.soainstitute.org
  - www.service-architecture.com
- General XML news site: www.xml.com
- W3 Schools Tutorial Web site: www.w3schools.com

Lab Exercise: Web services Overview Lab
Summary

- SOA is an architectural style, otherwise known as an architectural pattern.

- A business service (or simply service) is defined as a distinct, self-contained (or sealed) unit of automation logic that provides a specific business process.

- Conceptually, SOA is quite simple. SOA is based on software serving one of three roles:
  - Provider of a service – a business service/a unit of automation logic.
  - Requestor of service(s) – also known as a consumer or client.
  - Registry (or broker) of service(s) – a manager of a list of business services for requestors.

- SOA is about building business services that conform to a set of specific characteristics, or as some say, adhere to certain SOA principles.

- SOA assumes no particular technology or standard. It is just a style with a prescribed set of architectural principles.
  - Web services put SOA into practice.
  - Web services are SOA with discipline and structure.

- A Web service is unit of automation logic that receives messages and responds to those messages.
  - The messages are typically sent through the Web using HTTP (the transport protocol).
  - The messages are typically formatted in XML.
  - The messages typically follow the SOAP standard (the message protocol).
  - The services are described using a Web Service Definition Language document.
  - Potential consumers of a service can find Web services using a registry following the Universal Description Discovery Integration (UDDI) standard.

- The W3C, OASIS, and WS-I are all organizations working to define and manage standards that define Web services.