Service Oriented Architecture: Basics and Best Practices.

Dr. Srinivas Padmanabhuni
Principal Researcher
SOA Centre of Excellence, SETLabs,
Infosys Technologies Ltd, Bangalore, India
Srinivas_p@infosys.com
Agenda

• SOA Defined
• SOA as evolution of distributed computing models
• Our view of Enterprise SOA
• SOA and Web services
• SOA Standards Stack
• Loose Coupling via Document Style Web services
• Service Oriented Design and Development
• SOA Adoption Dimensions
• SOA Myths
• Best Practices in SOA Adoption
  – Overall
  – Phase Wise
• Conclusions
Service-Oriented Architecture Defined

Service-oriented architecture (SOA) is an approach to distributed computing:

- that is loosely coupled,
- that is protocol independent,
- that is standards-based,
- where software is accessed as a service,
- where software resources interact over a network according to a contract
Defining a service

A service

• Provides contractually defined behavior
• Can be implemented and fulfilled by any component
• Can be used by any component, based solely on the contract.
• Is coarser in granularity than a component and finer than an application
The SOA stakeholders

Service Consumer
- locates the required service, and all information for binding/invoking the service, from the Service Broker

Service Broker
- maintains a registry of services, their interface descriptions, provider information and invocation methods

Service Consumer
- finds and discovers the required service

Service Broker
- registers the service details

Service Contract
- stipulates the conditions and terms under which a service is provided

Service Provider
- provides service functionality that is published by the Service Broker

Client
- binds and invokes the service
SOA – The next in architectural evolution

<table>
<thead>
<tr>
<th>Tight</th>
<th>Loose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine</td>
<td>Coarse</td>
</tr>
<tr>
<td>Coarse</td>
<td>Enterprise</td>
</tr>
</tbody>
</table>

**Object Oriented** (for e.g. Java Objects)
- Objects implement fine grained functionality
- Objects are accessed in memory
- Base unit – Objects

**Component based** (for e.g. EJB’s & .NET remoting)
- Defines well defined functionality from a set of objects
- Components are accessed via distributed middleware (RMI, .NET remoting)
- Base unit – Classes, jars, wars

**Service Oriented** (for e.g. web services)
- Represents a higher-lever business function
- Predominantly asynchronous and stateless
- Services are accessed via the network (HTTP etc.)
- Base unit - schemas/interfaces
Comparing pre-cursors of Web services to Web services

- DCOM requires all participating nodes in a distributed application to be running a flavor of Windows and thus, is more proprietary in nature.
- CORBA ORB’s from different vendors can interoperate however cannot interoperate with DCOM
- CORBA and DCOM-Interface Definition Language (IDL) is much less expressive
- CORBA and DCOM are all based on binary protocols
- CORBA IDL cannot accommodate attachments such as DOC or PDF files as part of the message.

- The participating nodes can belong to any technology or running on any platform.
- Interoperability is not only message body level but also includes security, management etc. Messages are interoperable across all vendors.
- Expressive XML-schema makes integration in business/financial applications much easier
- SOAP is text-based and optionally includes type info as part of the message, which simplifies debugging and traffic monitoring since the message content is human-readable.
- SOAP allows MIME attachments as part of the message content.
But RPC Style Web services still retain the conventional distributed computing style
Loose Coupling by breaking the RPC tradition

Document Style Message Exchange
Our view of the target architecture of an ideal service oriented enterprise
## Key Aspects of a Service Oriented Enterprise

### Elaborated

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key to success in SOA lies in codifying and precise specification of business processes to enable automation</td>
<td></td>
</tr>
<tr>
<td>Infrastructure tier could be in the form of a utility computing model providing IT functionality on tap</td>
<td>The Integration Tier could be handled by an Enterprise Service Bus, MOM, or conventional EAI products</td>
</tr>
<tr>
<td>Web Services - key enablers for a Service Oriented Enterprise</td>
<td></td>
</tr>
<tr>
<td>Integration Tier should handle the service management as well as service lifecycle management</td>
<td></td>
</tr>
<tr>
<td>Business Processes Drive Enterprise Architecture</td>
<td></td>
</tr>
<tr>
<td>Service Interface -</td>
<td></td>
</tr>
<tr>
<td>• implementation independent</td>
<td></td>
</tr>
<tr>
<td>• ideally coarse grained</td>
<td></td>
</tr>
<tr>
<td>• defined by Business process owners</td>
<td></td>
</tr>
<tr>
<td>Business Services take part in business processes</td>
<td></td>
</tr>
<tr>
<td>Benefit of having a business service tier independent of underlying service management infrastructure is that the underlying implementation can be changed keeping the service interface intact</td>
<td></td>
</tr>
</tbody>
</table>
Web Services – an implementation of SOA

Loosely coupled standards based implementation of SOA
Web Services – a definition

A Web Service is a unit of software that:

- Processes **XML messages** framed using **SOAP**
- Describes its messages using **XML Schema**
- Provides an interface description using **WSDL**
- Can be discovered using **UDDI (optional)**
- Is transport independent (HTTP/JMS/SMTP…)

These are not web services (though they may qualify as services) ….

- XML over HTTP (or any other transport)
- XML over MQ/JMS
Web Services Core Standards Stack

- Standard discovery and registry - UDDI
- Standard Descriptor - WSDL
- Standard Messaging - SOAP
- Standard Data Types – XML Schema
- Standard Language - XML
Web Services in Action with Core Standards

1. Agree on Semantics (XSD)

2. Input Semantics (XSD)

3. Get WSDL (Optional)

4. Interact (SOAP)

Source: W3C Web Services Architecture Group

Requester Entity
- Requester
- Requester Agent

Provider Entity
- Provider
- Provider Agent

Discovery Agent (UDDI) (Optional)

Find WSDL (Optional)

Publish WSDL (Optional)

Source: W3C Web Services Architecture Group

Win in the flat world
SOA Standards Stack
SOA with Web Services - standards stack

Management
WS-Manageability

Presentation
WSRP

BPEL4WS, BPML
Business Process Orchestration
XLANG, WSFL, WSCI

Security
WS Security
XML Encryption
XML Signature

XKMS, SAML,
XACML, WS-Trust,
WS-Federation

Reliable messaging
WS-Reliability
WS-Reliable Messaging

Transaction
WS-Coordination, WS-AtomicTransaction, WS-BusinessActivity

WS-CoordinationFrmk, WS-
TransactionMgmt

Security

Transaction

Provisioning

Addressing
WS-Addressing

Discovery and Registry
UDDI, WS-Discovery

Description
WS-Policy, WS-PolicyAttachment

WSDL, WSFL

Messaging
SOAP, XML, XML Schema

HTTP, HTTP(s), JMS, SMTP...

Transport

Core standards
MSCF/IBM
OASIS
Other

Win in the flat world
Web Services – accepted standards stack

- Management
- Presentation
- Business Process Orchestration
  - BPEL4WS
- Security
  - WS Security
  - XML Encryption
  - XML Signature
  - XKMS, SAML
- Reliable Messaging
  - WS-Reliable Messaging
- Transaction
- Provisioning

- Core standards
- MSFT/IBM
- OASIS
- Other

- WS-Addressing
- Addressing
- Discovery and Registry
  - UDDI
- Description
  - WS-Policy, WS-PolicyAttachment
- Messaging
- Transport
  - SOAP, XML, XML Schema
  - HTTP, HTTPS, JMS, SMTP...
XML

- XML stands for E XTensible Markup L anguage
- XML is a markup language much like HTML
- XML can be used to describe and Transfer data
- XML tags are not predefined. User must define his own tags
- XML uses a Document Type Definition (DTD) or an XML Schema Definition to describe the data
- XML with a DTD or XML Schema Definition is designed to be self-descriptive
XML Schema

- defines elements that can appear in a document
- defines attributes that can appear in a document
- defines which elements are child elements
- defines the order of child elements
- defines the number of child elements
- defines whether an element is empty or can include text
- defines data types for elements and attributes
- defines default and fixed values for elements and attributes

```xml
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.w3schools.com"
  xmlns:xs="http://www.w3schools.com" elementFormDefault="qualified">
  <xs:element name="intro">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="first_name" type="xs:string"/>
              <xs:element name="last_name" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="address">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="city" type="xs:string"/>
              <xs:element name="state" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
What is SOAP?

- Is XML based stateless, one-way message exchange paradigm for exchanging structured and typed information in a decentralized, distributed environment.
- Provides the framework by which application specific information may be conveyed in an extensible manner.
- Applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information.
- Is silent on the semantics of any application-specific data it conveys, as it is on issues such as the routing of SOAP messages, reliable data transfer, firewall traversal, etc.
- Skeleton of SOAP contains a SOAP Envelope with an optional SOAP Header element and compulsory SOAP Body element.
## What is WSDL?

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

### Constituents of WSDL

<table>
<thead>
<tr>
<th>Constituent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Types</td>
<td>a container for data type definitions defined in schema</td>
</tr>
<tr>
<td>Messages</td>
<td>an abstract typed definition of the data being communicated</td>
</tr>
<tr>
<td>Operations</td>
<td>an abstract description of the action supported by the service</td>
</tr>
<tr>
<td>Port Type</td>
<td>an abstract set of operations supported by one or more end points</td>
</tr>
<tr>
<td>Binding</td>
<td>a concrete protocol and data format specification for a particular port type</td>
</tr>
<tr>
<td>Port</td>
<td>a single end point defined as a combination of a binding and a network address</td>
</tr>
<tr>
<td>Service</td>
<td>collection of related end points</td>
</tr>
</tbody>
</table>
What is UDDI?

- An UDDI implementation that provides the ability for a potential service consumer to determine the identity and connection details of available services.
- The UDDI server provides the relevant details for specific services categorized on the basis of industry, service type etc.
- The UDDI API provides the ability to search for specific services based on functionality.
- Adoption of UDDI is likely to be at an intra enterprise level initially.

The UDDI implementation enables a business to:

- Describe its business and its services.
- Discover other businesses that offer desired services.
- Integrate with services provided by other businesses.
Soap Request/Response for Java Service

**Request**
- POST /axis/services/Sum HTTP/1.1
- Content-Length: 424
- Content-Type: text/xml
- SOAPAction: calculateSum
- Host: localhost
- User-Agent: Java/1.4.1_02
- Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
- Connection: keep-alive

```xml
  <arg0 xsi:type="xsd:int">25</arg0>
  <arg1 xsi:type="xsd:int">25</arg1>
</calculateSum></soapenv:Body></soapenv:Envelope>
```

**Response**
- HTTP/1.1 200 OK
- Content-Type: text/xml; charset=utf-8
- Transfer-Encoding: chunked
- Date: Sat, 12 Jun 2004 11:07:49 GMT
- Server: Apache Coyote/1.0

```xml
  <soapenv:Body>
    <calculateSumResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <calculateSumReturn xsi:type="xsd:double">50.0</calculateSumReturn>
    </calculateSumResponse>
  </soapenv:Body>
</soapenv:Envelope>
```
Loose Coupling via Document Style Web services
RPC Style Web Services

RPC Web services present a distributed function (or method) call interface that is familiar to many developers. Typically, the basic unit of RPC Web services is the WSDL operation.

```xml
<wsdl:message name="getNamesSoapIn">
  <wsdl:part name="strName" type="s:string" />
</wsdl:message>
<wsdl:message>
  <wsdl:part name="getNamesSoapOut" type="s:string" />
</wsdl:message>
<wsdl:message>
  <wsdl:operation name="getNames">
    <wsdl:input message="tns:getNamesSoapIn" />
    <wsdl:output message="tns:getNamesSoapOut" />
  </wsdl:operation>
</wsdl:message>
<wsdl:portType name="ServiceSoap">
  <wsdl:operation name="getNames">
    <wsdl:input message="tns:getNamesSoapIn" />
    <wsdl:output message="tns:getNamesSoapOut" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ServiceSoap" type="tns:ServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
  <wsdl:operation name="getNames">
    <soap:operation soapAction="http://tempuri.org/getNames" style="rpc" />
    <wsdl:input>
      <soap:body use="encoded" namespace="http://tempuri.org/">
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </wsdl:input>
  </wsdl:operation>
</wsdl:binding>
```
RPC Style Web Services is better avoided

- RPC is synchronous
- RPC is method invocation
- RPC creates inter-object style interfaces
- Marshaling and serialization overhead
- Implemented by mapping services directly to language-specific function or method calls
- Chattiness of invocations
Document Style Web Services

- Document Style Web Service are loosely coupled and the request / response are in the form of XML documents.
- The client sends the parameter to the Web Service as XML document, instead of discrete set of parameter values.
- The Web Service processes the document, executes the operation and constructs & sends the response to the client as an XML document.
Document Style Web services do not put restrictions on payload and hence expect processing on either side.
You get to decide what message comes and goes by changing the schema inside the contract.

```xml
<wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">  
    <s:element name="getNames">
        <s:complexType>
            <s:sequence>
                <s:element minOccurs="0" maxOccurs="1" name="strName" type="s:string" />
            </s:sequence>
        </s:complexType>
    </s:element>
    <s:element name="getNamesResponse">
        <s:complexType>
            <s:sequence>
                <s:element minOccurs="0" maxOccurs="1" name="getNamesResult" type="s:string" />
            </s:sequence>
        </s:complexType>
    </s:element>
  </s:schema>
</wsdl:types>

<wsdl:message name="getNamesSoapIn">
  <wsdl:part name="parameters" element="tns:getNames" />
</wsdl:message>

<wsdl:message name="getNamesSoapOut">
  <wsdl:part name="parameters" element="tns:getNamesResponse" />
</wsdl:message>
```
Document Style Web Services

• No direct mapping between the server objects and the values in XML documents.
• The protocol places no constraint on how that document needs to be structured.
• **Does not require a rigid contract**
• Better suited for asynchronous processing
Service Based Design Principles

- Service Boundaries are explicit
- Services Are Autonomous
- Services Share Schema and Contract, Not Class
- Service Compatibility Is Based on Policy

-The four tenets of SOA from Don Box
## Tenets of SOA Development Methodology

### Traditional Application Development Approach

- A software application is the main target for requirements, design, coding and testing,

- Primary focus is on the implementation (code)

- Integration with teams is tightly coupled

- No metadata considerations

### SOA approach

- Service is the first-class object here

- Primary focus is on the service contract (WSDL etc.)

- Integration with partners/teams is loosely coupled.

- Service metadata (policies, roles etc.) are critical

- Service deployment will need to take care of issues like versioning, change management etc. at an individual service level

- Agile Programming is one way to program services, but not necessarily the only one
SOA Application Domains

- Data Management
- Business Intelligence and Reporting
- Legacy enablement
- Stakeholder Collaboration
- Seamless Integration
- Enterprise Application Portfolio Rationalization
- Enterprise Architecture Transformation

Tracks

SOA Application Domains
SOA: An EAI Perspective realized through ESB

Service Flow → Existing Applications → Data → New Service Logic

Enterprise Service Bus (ESB)

- BPEL Workflow
- Transformation Engine
- Routing engine
- XML Tools (Schema, Validators, XPath evaluators)
- Messaging backbone

Portal

SOAP Service Request (e.g. J2EE, .NET)

B2B Interactions
• Data services provide connections to multiple data bases
• A single view of critical data mapped to individual tables
• Powerful querying mechanisms based on XQuery,
SOA Myths

• SOA = Web Services
• Wrapping Services over Applications yields services
• SOA is a technology issue
• Shared Semantics is trivial
• SOA needs fully standardized mechanisms
• SOA can be done at an application level
• SOA can be implemented just by bringing an ESB
Best Practices for SOA: Overall Adoption

SOA requires **strategic approach for maximum Reuse and flexibility**, with models for sharing Costs, benefits across the organization

Majority of implementations as we see today are **stopping at Integration+** owing to Governance issues

SOA **need not mean same thing** for everyone. Depending on the application domain it can be for EAI, Or for B2B integration or for even Infrastructure Virtualization

There is **Strong need to develop both Processes and People dimension** in addition to technology dimension

Implementing SOA is **not just about implementing an ESB** as product vendors may claim

**Architects and Developers need to retrained** to follow contract first development approaches

Initial investment is high, with payoff in the longer run, hence a **long term view is important**
Best Practices in SOA Adoption Phase wise (1/3)

**Information Elicitation**

- Understanding of the business and IT landscape and identifying the drivers are essential in building a strong case for service adoption.
- Understanding the scale to which the service adoption has to cover. Usually certain lines of business might be very crucial to adopt services.
- Going through the SOA assessment to check the readiness is very important. It should include operational readiness, business and IT resources necessary to support the effort. The primary objective of this should be to highlight the gaps in relation to the reference SOA model.

**Service Identification**

- For successful SOA adoption clear understanding on service enabled business model is essential.
- Perform top down service identification by domain decomposition of existing and desired reusable business processes.
- Defining business process as service level interaction and constant validation of services being aligned with the business process improvement effort is essential.
- Tweaking of the internal processes to suite the service enabled business model must be considered.
- Consolidation of identified services to define and preparing services catalogue.
- Setting up a working group that would come up the charter on what needs to be achieved and what would be the success factors.
Best Practices in SOA Adoption – Phase wise (2/3)

Service Definition

- Defining interface, data model, assertions and obligations for the service
- Models for conversation, message exchange patterns, sequencing and QoS Tweaking of the internal processes to suite the service enabled business model must be considered
- Identification of essential service properties/meta-data
- Identification of NFR for services such as security, validations, logging, monitoring etc.

Service Realization

- A master plan needs to be in place before starting on the SOA exercise, this should discuss the work plan, risks, mitigations, communication etc
- Choosing right tools and technology for services creation, service contract definition and communication of the same to involved stakeholders.
- Choosing right tools and technologies for service integration and orchestration.
- Choosing right service invocation and execution approach for a particular scenario
- Defining transformation strategies involving both invasive and non-invasive approaches.
Service Institutionalization

- Continuously keep researching on the new trends, standards and adopt into the SOA exercise as necessary after piloting and measuring their impact.
- A common understanding on the glossary of terms that would be regularly used all throughout the SOA exercise.
- Have a conceptual service architecture model that gives the relations between various architectural entities that make up the SOA.

Service Governance

- Clearly define the roles and responsibilities, ownerships, stakeholders associated with the program, which should be integrated with the governance.
- A governance model and policy should be set up that should provide roadmap, policy management, communicate the decisions etc.
- The governance body should be responsible in setting up the service management framework, define the support structure, delivery, versioning etc.
- Clearly defining service consumption/invocation ownership at service consumer end, rather delegating service consumption at centralized point such as ESB.
Conclusions

• SOA enables a loosely coupled model of distributed computing
• SOA is best applied with strategic intent on the scale of an enterprise
• A structured approach to SOA planning is vital
• SOA is not the same as web services though web services are the most popular means of achieving SOA
• SOA needs to looked at from the domain of its application
• Best practices right through life cycle of SOA planning will yield optimal results