The Evolution of Service-Oriented Development

Bill Jones
Oracle Corporation
bill.jones@oracle.com
Agenda

- What’s important about SOA
- Where we are
- How did we get here
- Where are we going
What’s Important about SOA

- Enables IT agility
- Reduces cost of integration
- Enables new types of applications
Where Are We?

- Standards
  - SOAP, WSDL, XML, Schema, UDDI, WS-I 1.xSwA
  - WS-Security, WS-ReliableMessaging

- Tools
  - Provide service enablement

- Solutions
  - Process orchestration, BPM
  - Integration, ESB
  - Service Management, WSM
## Java Standards

<table>
<thead>
<tr>
<th>Java APIs for XML</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAXP</td>
<td>Java API for XML Parsing</td>
</tr>
<tr>
<td>JAXB</td>
<td>Java API for XML Data Binding</td>
</tr>
<tr>
<td>JAX-RPC</td>
<td>Java API for XML Remote Procedure Call</td>
</tr>
<tr>
<td>SAAJ</td>
<td>SOAP API for Attachments in Java</td>
</tr>
<tr>
<td>JAXR</td>
<td>Java API for XML Registries</td>
</tr>
<tr>
<td>EJB 2.1</td>
<td>Stateless Session EJB Endpoint Model</td>
</tr>
<tr>
<td>JSR 109</td>
<td>Web Services Deployment Model</td>
</tr>
</tbody>
</table>
Whence SOA?

- Inside technologies
  - RPC
  - Distributed Transaction Coordinators
  - Distributed Objects
  - MOM
  - Integration platforms
Whence SOA – Inside

Phase I
- Downsize
  - Client/Server
  - DCE

Phase II
- Components
  - CORBA
  - DCOM
  - EJB

Phase III
- Messaging
  - MOM
Common Characteristics

- Tight coupling
- Shared *implicit* assumptions
- Proprietary technology
- Suited for communication where you have control over both sides of interaction
- Extension of local programming model
- Generally more efficient, easier to develop and debug.
Whence SOA?

- Outside technologies (B2B)
  - EDI
    - “the computer-to-computer exchange of structured information, by agreed message standards, from one computer application to another by electronic means and with a minimum of human intervention” – Wikipedia
  - ebXML
  - Verticals
    - RosettaNet, OAGIS, etc.
Whence SOA – Outside
Common Characteristics

- Loose coupling
- Explicit semantics and metadata
- Standard internet technologies
  - Combined and/or leveraged in proprietary ways
- Attempt to address concerns of inter-organization interactions
  - Security, federation, trust, authorization, liability, etc.
Agenda Checkpoint

- What’s important about SOA
- Where we are
- How did we get here
- Where are we going
Customer Needs

Increasingly Demanding Users
- End-to-End Processes
- Shorter Change Cycles
- Better Insight and Auditing

Increasingly Complex Infrastructure
- Heterogeneous Systems
- Silos
Service-Oriented Architecture (SOA)

Web Solution

SOA Solution

1. More Interoperable
2. Agile Business Processes
3. Richer Clients
4. Improved Reuse
Customer Challenges

- Building a Business Case for SOA
- Selecting the Right Projects
- Building a SOA Roadmap
- Applying SOA Principles to Projects
- Realizing a Service-Oriented Solution
- Evolving Enterprise Architecture
- Affecting Organizational Changes
- Continuing to Succeed with SOA
Key Underlying SOA Enablers

- Technology
- Standards
- Strategy
- Architecture
- Information
- Governance
- Organization
- Processes
- Delivery
- Operations

Experiment and Learn

Get Maximum Benefits
Where Are We Going

- Core Standards evolve
  - SOAP 1.2, WSDL 2.0, XML 1.1, UDDI 3.0, WS-I additional profiles (e.g. BSP, RAMP)
  - WS-Security 1.1, WS-RX, MTOM

- Additional Standards Mature
  - WS-Trust, WS-SecureConversation, SAML 2.0
  - WS-Notification, WS-Transaction, more?
## Java Standards

<table>
<thead>
<tr>
<th>Java APIs for XML</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAXP</td>
<td>Java API for XML Parsing</td>
</tr>
<tr>
<td>JAXB</td>
<td>Java API for XML Data Binding</td>
</tr>
<tr>
<td>JAX-RPC</td>
<td>Java API for XML Remote Procedure Call</td>
</tr>
<tr>
<td>SAAJ</td>
<td>SOAP API for Attachments in Java</td>
</tr>
<tr>
<td>JAXR</td>
<td>Java API for XML Registries</td>
</tr>
<tr>
<td>EJB 2.1</td>
<td>Stateless Session EJB Endpoint Model</td>
</tr>
<tr>
<td>JSR 109</td>
<td>Web Services Deployment Model</td>
</tr>
</tbody>
</table>
Methodologies

- Emphasize the ‘A’ in architecture
- Top down vs. bottom up
- Interoperability as a design pattern
- Governance is not just a product
  - ➢ Internal standards, best practices
- Reference Architecture?
SOA: Top Down versus Bottom Up?

Business Analyst
- Activity → Activity → Activity
- Notation Layer
  - BPMN or UML

Integration Developer
- assign → invoke → receive → assign
- Executable Layer
  - XML, XQuery, BPEL, Rules

Service Developer
- Business Services
  - Adapters, Java, Struts, JSF
- Existing Systems
  - DATABASE, PACKAGED APPLICATIONS, JAVA, MAINFRAME
Where Are We Going

- **Solutions**
  - Composite Applications Frameworks
    - Service Component Architecture, JBI
  - SOA Governance
    - Metadata management, Policy Management, Development and interface standards, Automating and verifying best practices
  - B2B?
Example – Publish Inventory to Account Reps

Inventory

Warehouse Management

Portal
Example – Cashflow Management

- Business Activity Monitoring
  - Alerts
  - Manufacturing Schedule
  - Work in Progress
  - Manifests
  - RFID Events

- Manufacturing Execution
- Logistics
- RFID

Bill Jones — The Evolution of Service-Oriented Development
SCA in a Nutshell

- SCA models the “A” in SOA – for systems composed of reusable services
  - model for service-based system:
    - service construction
    - service assembly
    - deployment
  - Heterogeneity – supports components from
    - multiple languages
    - multiple container technologies
    - multiple service access methods
- 0.9 Level specification published in November 2005
SCA – High Level Points

- Unified declarative model describing service assemblies
  - dependency resolution and configuration
  - declarative policies for infrastructure services
    - Security, Transactions, Reliable messaging
- Business-level model for implementing services
  - service components with service interfaces
- Binding model for multiple access methods
  - WSDL, SOAP over HTTP
  - but also: JMS\textsuperscript{TM}/messaging, Java\textsuperscript{TM} RMI/IIOP...
  - Java\textsuperscript{TM} interfaces are good, as are WSDL portTypes
Service Assembly Model

- Unified, language-independent way to expose implementations as services
  - Java™, BPEL, PHP, other languages (including .NET)
- Technology independent modelling and composition of service networks
  - Service dependencies
  - Resolution through wiring
- Facilities for dynamic service configuration
  - Properties / Protocols / Qualities of service
  - Profiles
- Design time and deployment time configuration
Assembly Model Concepts

- Design Time Assembly
  - Composite
  - Implementation
  - Component
  - Service
  - Reference
  - Wire
Service Assembly Model

- Model for assembling tightly coupled code (Composites)
- Model for assembling loosely coupled services (Systems)
Composite

Service
- Java interface
- WSDL PortType

Composite A

Component A

Property setting

Component B

Reference
- Java interface
- WSDL PortType

Service

Wire

Wire

Wire

Binding
- Web Service
- SCA
- JCA
- JMS
- SLSB
- ...

Properties

Reference

Binding
- Web Service
- SCA
- JCA
- JMS
- SLSB
- ...

...
Composite

- **Assembly of services** developed and deployed together
- **Contains**
  - service implementations organized as *components*
  - required services as *references*
  - public *services*
  - *wires* connect components, services, and references
- **Used as implementation of components at next higher layer**
Implementations

- Basic elements of business function
- Support for different implementation technologies
  - e.g. Java™, BPEL, C++, PHP, ...
  - implementation type extensibility
- Provides business function via one or more services
- Uses other services through service references
- Service and references typed by interfaces
  - remotable, bi-directional, companion callback interface
- Scoped
  - Runtime managed state and routing
Component

- *Configured* instance of implementation within a Composite
  - more than one component can use same implementation
- *Provides* and *consumes services*
- Sets *properties*
- Sets service *references* by *wiring* them to services
  - wiring to services provided by other components or by references of the composite
Represent *services* that are external to the composite
  - accessed by components within the composite like any other component service

Use *bindings* to describe the access mechanism to the external service
- *e.g.* Web service, stateless session EJB™, JMS™, JCA™, ...
- binding type *extensibility*
- overridable (no, may, must)

Can define required characteristics through *Policy*
Service

- Used to *publish services* provided by composite, for use by external clients of the composite
  - can be service provided by component or a reference
- Use *bindings* to describe the access mechanism that external clients have to use
  - Web service, stateless session EJB\(^\text{™}\), JMS\(^\text{™}\), ...
  - binding type extensibility
  - always overridable
- Can define provided characteristics through *Policy*
SCA Interaction Model

- Synchronous & Asynchronous service relationships
- Support for Conversational services
- Asynchronous support
  - “non-blocking” invocation
  - Callbacks
    - Point to point
  - Extended for business event support
    - Pub-sub
Example

```
bigbank.accountcomposite

Service
AccountService

AccountService Component

AccountData Service Component

Reference
StockQuote Service
```
SCA Composite Example

<?xml version="1.0" encoding="ASCII"?>
<composite xmlns="http://www.osoa.org/xmlns/sca/0.9"
    name="bigbank.accountcomposite">

    <service name="AccountService">
        <interface.bpel process="AccountService"/>
        <binding.ws port="http://www.bigbank.com/AccountService#
wSDL1.endpoint(AccountService/AccountServiceSOAP)"/>
        <reference>AccountServiceComponent</reference>
    </service>

    <component name="AccountServiceComponent">
        <implementation.java class="services.account.AccountServiceImpl"/>
        <property name="currency">EURO</property>
        <reference name="accountDataService" target="AccountDataServiceComponent"/>
        <reference name="stockQuoteService" target="StockQuoteService"/>
    </component>

    <component name="AccountDataServiceComponent">
        <implementation.java class="services.accountdata.AccountDataServiceImpl"/>
    </component>

    <reference name="StockQuoteService">
        <interface.java interface="services.stockquote.StockQuoteService"/>
        <binding.ws port="http://www.quickstockquote.com/StockQuoteService#
wSDL1.endpoint(StockQuoteService/StockQuoteServiceSOAP)"/>
    </reference>
</composite>
Demonstration

SCA
What Is JBI?

- Unified Metadata (WSDL)
- Unified Addressing (Logical Naming)
- Plug-and-play Binding Components
- Optimized Transport and Data Format
- Both Sync. and Async. Interactions
JBI Advantages

- Core design is made for service oriented service provision and consumption
- Encapsulation of technical binding and technology
- Introduce meanings for standard installation and artifact deployment
- Introduce a standard way for managing components
- Support for mediated XML messaging (unlike JCA that is designed to provide EIS connectivity)
- Avoid Vendor Lock-in
Absolutely Shameless Self Promotion

- Co-author of **Java Transaction Processing: Design and Implementation**

- From amazon.com reviews:
  - “Required Enterprise Transactions Reading”
  - “All J2EE developers should read this book”
QUESTIONS & ANSWERS
Agenda

- Overview
- RPC
- SOA
- Web Services
- Integration and Orchestration
- Composition
Integration

- Technology
  - BPEL Process Orchestration
  - Enterprise Service Bus
  - Registry
  - Web Services Management & Security

- Standards
  - BPEL, WSIF, JMS, JCA
  - UDDI
  - WS-Security
Business Process Integration

Submit Order → Transform Order/Customer → Determine Discount

Check Inventory

Can Fulfill?

Formulate Fulfillment Offer

Formulate Rejection

Send Result to User

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?

Can Fulfill?
Building Business Processes

How do I …

- Coordinate asynchronous communication between services
- Correlate message exchanges between parties
- Implement parallel processing of activities
- Manipulate/transform data between partner interactions
- Support for long running business transactions and activities
- Provide consistent exception handling
Requirements for Processes

- **Process language**
  - Business Process Execution Language

- **Process runtime environment**
  - High performance execution of long running processes

- **Process monitoring and lifecycle management**
  - Management environment

- **Process design time environment**
  - Design time tools to visualize process development
The BPEL Alternative

Getting to Process Standards

- Build your own 1-off orchestration
- Traditional EAI
- The BPEL Alternative
  - Rich metadata (XML Schema, WSDL)
  - Sophisticated binding framework (WSDL)
  - Internet scale connectivity
  - Support for both sync and async interactions (BPEL)
  - Sophisticated flow coordination (BPEL)
  - Advanced exception management (BPEL)
  - Extensible
  - Composable
  - Convergence of application development and integration
<process>
  <!- Definition and roles of process participants -->
  <partnerLinks> ... </partnerLinks>
  <!- Data/state used within the process -->
  <variables> ... </variables>
  <!- Properties that enable conversations -->
  <correlationSets> ... </correlationSets>
  <!- Exception handling -->
  <faultHandlers> ... </faultHandlers>
  <!- Error recovery – undoing actions -->
  <compensationHandlers> ... </compensationHandlers>
  <!- Concurrent events with process itself -->
  <eventHandlers> ... </eventHandlers>
</process>
BPEL Activities

Process Programming Language

**Primitive Activities**
- `<invoke>`
- `<receive>`
- `<assign>`
- `<reply>`
- `<throw>`
- `<terminate>`
- `<wait>`

**Structured Activities**
- `<sequence>`
- `<switch>`
- `<pick>`
- `<flow>`
- `<link>`
- `<while>`
- `<scope>`
Example – Synchronizing Customer Information

INTEGRATION FLOW

Siebel CRM

Oracle Financials
Example: Managing Services & Applying Policies

Consumers
- Web
- Phone
- Systems
- Business Partners

Services
- Packaged Applications
- Fulfillment Center
- Customer Service

1. Apply Policies
- Security
- Monitoring
- Logging
- Auditing
Demonstration
Integration and Orchestration
Agenda

- Overview
- RPC
- SOA
- Web Services
- Integration and Orchestration
- Composition
Agenda

- Overview
- RPC
- SOA
- Web Services
- Integration and Orchestration
- Composition