Service-Oriented Architecture and Best Practices

Don Adams
Rourke McNamara
TIBCO Software, Inc.
Agenda

- **What is SOA**: 15 minutes
- **Architecture**: 1 hour
- **Standards**: 45 minutes
- **Best practices**: 45 minutes
- **Q&A**: 30 minutes
Services: The Big Picture

- **What is a service?**
  - A commonly used unit of functionality
    - e.g. Sales Order Management
  - Packaged for easy access and consistent re-use
    - Becomes a de-facto standard in the enterprise

- **Services are all about reuse and ROI**
  - Avoid re-inventing the wheel on the next project
    - ROI from IT cost reduction
  - Assemble new business processes from existing services
    - ROI from faster time-to-market – a more responsive enterprise

- **The functionality underlying most services already exists**
  - We’re making it more accessible
  - Integration is essential

- **It takes discipline and governance to get the ROI from services**
  - To build the right services and ensure they are reusable
  - To ensure that they actually get used
  - To operate and evolve the services once deployed
The Enterprise Challenges with SOA

Productivity
- Deliver more functionality
- In less time
- At a lower cost

Control
- Governance
- Performance
- Reliability

Integration
- Connect new applications to existing IT investments

SOA
IT Pain Points Addressed by SOA

Q: What are the IT/technology problems your company hopes to address using SOA?
Base: 521 (Among qualified respondents)

Source: InfoWorld Research Report: Service Oriented Architecture (SOA), March 2006
## Top 3 Drivers for SOA Segmented by Industry

<table>
<thead>
<tr>
<th>Business Drivers</th>
<th>Financial Services</th>
<th>Manufacturing</th>
<th>Transport &amp; Comm</th>
<th>Utilities</th>
<th>Retail / Wholesale</th>
<th>Services</th>
<th>Healthcare Services</th>
<th>Gov’t &amp; Education</th>
</tr>
</thead>
<tbody>
<tr>
<td>Competitive advantage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revenue growth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal operational / process efficiency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External process efficiency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IT Drivers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support dynamic business environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimal use of IT resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upgrade company’s IT infrastructure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long term cost savings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short term cost savings</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: IDC “SOA-Based Services Buying Trends: A 2006 Survey of U.S. Companies” May 2006 (sampling = 447 qualified respondents weighted by size of company)
SOA Speeds Time to Market

- SOA leverages existing application components instead of writing and testing new code

- This can shorten the length of the average application development project – perhaps as much as 75%

"More than 75% of enterprise applications in production are monolithic. Their business logic is not externally accessible in a modular form for easy reuse in other applications."

Gartner Integration Conference, May 2004

Source: Tutorial: How to Build a Cost Model for Service-Oriented Development of Applications, Michael Blechar, Gartner Application & Web Services Summit, April 2005
Service-oriented development of applications (SODA) is estimated to reduce total IT expenses over the long term by as much as 20% compared to traditional client/server development methods.

Savings becomes exponential over time as library of business services expands and greater degree of reuse is achieved.

Analysis compares cost savings using three types of Service-Oriented Development of Applications (SODA) when compared to traditional client/server application development:

- RAD = Rapid Application Development
- AMD = Architected Model-Driven
- ARAD = Architected Rapid Application Development

Source: Tutorial: How to Build a Cost Model for Service-Oriented Development of Applications, Michael Blechar, Gartner Application & Web Services Summit, April 2005
SOA Enables a New Breed of “Recomposable” Applications

SOA creates an architecture style that enables you to compose applications out of reusable services.

The ROI goes beyond IT cost reduction:

- Increased reuse,
- Greater flexibility, and
- Greater productivity

\textit{yield}

- Faster time to market!
Integration is an Essential Ingredient – SOA is Heterogeneous and Distributed

Application Platforms Used for SOA

- Java: 61%
- .NET: 60%
- J2EE: 54%
- C/C++: 24%
- COBOL: 8%
- Other Mainframe: 8%

The Challenge of Deploying Across Application Platforms

- Extremely Difficult: 16%
- Somewhat Difficult: 73%
- Not at all Difficult: 8%
- Don’t Know: 3%

Source: InformationWeek 2006 Survey
But We Need Governance to Realize Benefits:
24% of Companies say SOA Fails to Meet Expectations

The Reasons Given Why SOA Fell Short

- Introduced more complexity into IT system: 55%
- Cost more than expected: 41%
- Failed to provide expected level of integration: 35%
- Needed standard just emerging or not yet on drawing boards: 34%
- Too expensive to integrate legacy systems: 24%

Source: InformationWeek 2006 Survey
So What Does it Take to Succeed?

- Architecture
- Organization and Governance
- Standards
- And of course ... Best Practices!
SOA Architecture
A Business Process Built with Services
Services are More Than Operations!

- The responsibilities of a service often extend beyond a single operation

- The *Order Management Service* keeps track of the full life-cycle of the order

- Its operations step the order through its life-cycle
  - The *Submit Order Operation* creates the order
  - The *Update Status Operation* allows other parts of the business process to update the order status
Pure Request-Reply Operations Are Not Enough!

- When we order on-line, the goods do not arrive while we are sitting at the computer (most of the time, at least)!

- There is service activity after the order acknowledgement is returned
  - E.g. the release of the order to the warehouse for fulfillment

- The service can generate asynchronous responses!
  - E.g. status updates

- Other operations interact with work-in-progress
  - E.g. get order status, update status, etc.
Event Notification Decreases Coupling

- **In request-reply, the requestor specifies the operation to be performed**
  - Requires the requestor to know what operations need to be performed by which participants

- **Notification provides a design alternative**
  - The “Notification Service” announces that an event has occurred
    - i.e. the order has shipped
  - The “Subscriber” to the notification service decides on its own what to do when the notification is received
    - i.e. the Accounts Receivable Service sends an invoice
Typical Service Architecture

- **Services provide ease of access by standardizing:**
  - Access technology
  - Data semantics (common data model)
  - Operation semantics

---

**Native Interface**
- Native semantics for operation and data
- Native technology for operation and data

**Service Interface**
- **Technology of access**
- **Data semantics**
- **Operation semantics**

**Some level of standardization**
- Native semantics for operation and data
- Native technology for operation and data

**Service Approach**
- **Using Component**
  - Native Interface
  - Provider of Functionality

**Traditional Object/Component Approach**
- **Using Component**
  - Native Interface
  - Provider of Functionality
Not Every Service Needs Full Standardization

- **Standard TIBCO adapters ease access to legacy applications**
  - Standardize access technology
  - Native semantics for data structures and operations
  - Relies on communications infrastructure for security aspects

- **XML Common Data Model over RV or JMS**
  - Standardize access technology and data structure semantics
  - May or may not standardize operation semantics
  - Relies on communications infrastructure for security aspects

- **Full-blown SOAP**
  - Standardize access technology, standardize data structure and operation semantics
  - Provides security aspects independent of communications infrastructure
    - Requires a security infrastructure, however!

*We need to be clear about the standardization goals for each service*

*It is valid to have different standardization goals for different services!*
Infrastructure Services

- Building blocks that provide commonly required functionality in a standardized way
- Exposing lower level functionality as services significantly reduces the level of effort required to build higher level services
- Common infrastructure services include:
  - Messaging Services
  - Event Services
  - Audit and Logging Services
  - Error Notification Services
  - Security Services
  - Portal Services
- These are typically not SOAP services!
  - They are used to implement other services
Complete Uniformity is Not Always Possible

- Sometimes a re-usable component (library) needs to be provided in the user’s technology and embedded

- Examples:
  - Local interface for error logging
  - Security intercept for access control
Standardize Service Transports

- Select transports thoughtfully - these will become your standards
  - Mistakes will be expensive to correct

- Don’t expect to get to a single standard!
  - A single standard may be counter-productive
  - You get the ROI from minimizing the variations

- Common pattern: HTTP for interactions outside the enterprise, JMS within the enterprise
Choose Your Service Transports Carefully

Use (SOAP over) HTTP:
- When HTTP is already used or the accepted standard. Don’t reinvent the wheel unless necessary
- When zero client install is a requirement (e.g. portals or Web clients)
- For external services when installing JMS clients is not an option

Use (SOAP over) JMS:
- For asynchronous messaging
- For publish/subscribe
- For intermittently connected systems or devices
- For exactly-once delivery (e.g. mission-critical applications)
- For high volume and scalability

Consider JMS as the default bus for internal application communication
**Point-to-Point Services**

- **Point-to-Point Services** standardize the *technology* used to access operations and represent data.

- **Point-to-Point services do not** standardize the *semantics* of the operation or the data.

```
Using Component

Service Interface

Data or Application Service

Native Interface

Provider of Functionality
```

- Native semantics for operation and data
- **Standard technology for access**
Business Services

- Business services standardize both the semantics and the access technology
- The standardization greatly simplifies the reuse of the functionality in many contexts
- This standardization also makes it easier to construct or modify composite business services
Service Composites

- Service composites orchestrate the use of services to achieve some goal

- The composite may, itself, be a service

- A composite may be a complete business process, giving us Business Process Management
  - BPM and Business Works can be viewed as tools for creating composites

- Service composites make possible the overall management of the encapsulated business process including monitoring and error reporting
What Secondary Functions Might a Service Provide?

- Control over who has access (authorization)
- Validation of the user’s credentials (authentication)
- Accounting regarding who is using the service
  - Audit trail
  - Bill-back for service utilization
- Encryption/Decryption (only half!)
- Non-repudiation
- Performance tracking
  - Rates of utilization
  - Response time

Different services may require different levels of secondary functions!
Services Are Not Free!

- **More work at design time**
  - Increased levels of documentation and testing are required

- **Access overhead**
  - Granularity of work must outweigh overhead
  - We don’t make addition into a service
    - There are other ways to obtain reuse (e.g. shared code libraries)

- **Extra work at run-time**
  - Authentication, authorization, encryption, accounting, audit trails

- **Every interface shouldn’t be a service!**
  - Must demonstrate potential for reusability (i.e. ROI) to justify
    - Identify the multiple users of the service
    - Make sure that the functionality is, indeed, the same!
Specifying Services

- **Common functionality used in several use cases can be packaged as a service**

- **The operations of the service should be documented as distinct use cases**
  - Same use case definitions apply
    - Asynchronous event trigger (now indirect, via business use case trigger)
    - Trigger initiates a chain of activity
    - Activity produces a countable result

- **Ex: Consider “Obtain Disbursal Authorization”**
  - The transfer funds use case also needs this capability if the funds transfer is between banks
  - Define “Obtain Disbursal Authorization” as an implementation use case
ATM Example Services

- Bank Customer
  - Withdraw Cash
    - Obtain Disbursal Authorization Service
  - Transfer Funds
    - Report Funds Delivered Service
  - Make Deposit
    - Deposit Funds Service
Scenario Showing Service Design in Context

Customer: Person
- insert card and enter PIN
- select "Withdraw Cash"
- enter amount
- remove cash
- remove card and receipt

ATM Machine
- validate PIN
- prompt for transaction
- prompt for amount
- invoke obtain disbursal authorization
- Success?
  - Yes
    - Dispense Cash
  - No
    - print receipt and return card

ATM Server
- invoke report funds delivered service
- Determine Bank and Forward
- (disbursal request)
- (disbursal authorization)
- (forwarded request)
- (forwarded notification)
- (notification acknowledgement)
- (disbursal authorization)
- (related notification)

Bank
- Determine Bank and Forward
- grant disbursal authorization
- record withdrawal transaction
- (forwarded notification)
- (related notification)
The disadvantage of this approach is that you lose the big picture of how all the components interact to carry out the function.
**Service Design Specification**

- **Each overall service should be characterized by specifying:**
  - The abstraction level
  - The unit of work life-cycle the service encapsulates
    - All the states an order can go through
  - The operations provided by the service
    - Place order, modify order, cancel order, get order status
  - The data the service actively manages
    - What was ordered, shipping address, payment terms, etc.
  - Ancillary data required by the service
    - Customer information, product information
  - The relationship between the service and lower-level services on which the service depends
  - Key Performance Indicators (KPIs) and Service Level Agreements (SLAs)
  - Audit requirements for the service
Service Operation Specification

- Each operation of the service should be characterized by specifying:
  - The operation interface
  - The semantics of the operation being invoked.
    - What changes does it make that are visible?
  - The input and output data structures and their semantics
  - Performance requirements
  - The required availability of the operation
  - The invocation style for the service
    - On-demand, event driven, continuous
  - The coordination approach used
    - Fire-and-forget, request-reply, etc.
  - Authentication required
  - Authorization required
  - Data protection (encryption) required
You Will Never Build a Perfect Service!

- Be happy if you get close enough that the changes are minor
  - E.g. adding a field here or there

- Plan for service evolution
  - Infrastructure must allow the simultaneous deployment of both old and new service versions
  - Service users can gradually convert to the new version
The Reusability Challenge

- How do we design for future usages?
  - Today we enter orders in person, via paper, by phone, on-line, …
    - What’s next – via Blackberry? Automatic re-order?
  - Your CPG firm decides to sell branded clothing as a promotion!
    - Orders now need sizes, colors, etc.

- Insight is required when conceptualizing a service
  - What might change in the future?
    - Evolutionary changes – organic growth
    - Revolutionary changes – buying your biggest competitor, new markets
  - How do these changes challenge existing functionality?
  - Which alternatives are worth investing in?

Who can provide this insight in your organization?
Business and Infrastructure Services Differ!

- **Infrastructure Services encapsulate portions of system processes**
  - E.g. “Report error” or “Make Audit Entry”
  - The requirements come from the technical community
  - The users are future technical projects

- **Business Services encapsulate portions of business processes**
  - e.g. “Place Sales Order” or “Invoice Customer”
  - The requirements come from the business community
  - The users are future business processes

_Different organizations are involved!_
The Silo Problem – Who Owns the Big Picture?
Business Service

- Application silos include both the business and IT sides of the house

- New SOA technology silos generally have no business counterpart, and yet:
  - We expect them to build reusable business services
  - We expect them to orchestrate business services to build business processes and composite business services

These expectations are not realistic without business participation!
Who defines the service? Who pays for it?
Potential Business Services are Discovered in Projects

- Functionality in a business process is identified by the project team as being potentially reusable

- Scope of reusability needs to be determined
  - Project team likely is narrowly focused on project needs
  - Where does broader perspective come from?
  - Who funds the effort to broaden the functionality into a service?

Who evaluates business service proposals?
Services Steering Committee(s)

- **Steering committee membership**
  - Senior Business Process Architects
  - Senior Systems (Software) Architects

- **Centralized approach**
  - One organization evaluates all service proposals and specifies services
  - Limited in scalability – potential bottleneck
  - Appropriate for small-to-mid-size companies with centralized development organizations
    - e.g. Harrah’s

- **Silo Approach**
  - Each major silo (application area) has a steering committee consisting of
    - Representatives of the silo itself
    - Representatives of the silos that need to interact with it
    - Enterprise architects (business process and systems)
  - Appropriate for large companies with distributed development
    - PepsiCo, FedEx, Citigroup

- **A process must be in place to:**
  - Engage the steering committee
  - Determine who will implement the service
  - Determine who will absorb the additional costs (there WILL be extra cost!)
What Does the Steering Committee Do?

- **Identify usage scenarios for the potential business service**
  - As wide a variety as possible

- **Evaluate the business usage scenarios to determine:**
  - If the operations are truly the same
  - If the unit-of-work milestones are truly the same
  - If the non-functional requirements (KPIs and SLAs) are the same

- **If the usage appears the same, then specify the service**
  - This full utilization perspective is needed to make the service usable in all contexts
  - Failing to do this properly will result in a guaranteed re-work of the service the next time it is used!
Where Do Services Make Sense?

- When there is functionality that is either used in more than one place or is provided in more than one place, particularly when those “places” are different applications.
Service Evaluation Criteria

- **Potential reusability of the service in other contexts**

- **The simplicity and clarity of using the service:**
  - Easy-to-understand roles for services
  - Simple interactions with services
  - Simple orchestration of services
  - Clearly defined and auditable process milestones
    - Basis for SLA, KPI, Sarbanes-Oxley

- **Ability to monitor/manage the service**
  - Maintain high availability
  - Identify and handle problems before SLAs are impacted
  - Measure KPIs for six-sigma process improvement

- **Flexibility with respect to likely business changes**
  - Will the service still be usable in the face of likely changes?
Technical Challenges

Is it even possible to use the same service everywhere?
Governance is Essential

- **When creating a service**
  - To decide what ought to be a service
    - Ensuring ROI
    - Limiting speculative service development
  - To ensure those with appropriate insight participate in specifying the service

- **When designing applications**
  - To ensure information about services is appropriately disseminated to potential users
  - To ensure that services get used and not re-invented

- **When operating services**
  - To coordinate service operation with dependent systems
Organizational Best Practices for Projects
SOA-related Standards

Shivajee Samdarshi
Horizontal Architecture Group
Agenda

- Standards overview
- Criteria for standards participation
- Maturity model
- Web Services standards
- Messaging standards
- Other standards
- What are we leading?
- What are we learning?
- Conclusion
Webster’s Definition of “Standard”

Function: noun
Etymology: Middle English, from Old French estandard rallying point, standard, of Germanic origin; akin to Old English standan to stand and to Old English ord point

1 : a conspicuous object (as a banner) formerly carried at the top of a pole and used to mark a rallying point especially in battle or to serve as an emblem
2 a : a long narrow tapering flag that is personal to an individual or corporation and bears heraldic devices b : the personal flag of the head of a state or of a member of a royal family c : an organization flag carried by a mounted or motorized military unit d : BANNER
3 : something established by authority, custom, or general consent as a model or example : CRITERION
4 : something set up and established by authority as a rule for the measure of quantity, weight, extent, value, or quality
Webster’s Definition of “Proprietary”

Function: noun

1 : one that possesses, owns, or holds exclusive right to something; specifically: PROPRIETOR
2 : something that is used, produced, or marketed under exclusive legal right of the inventor or maker; specifically: a drug (as a patent medicine) that is protected by secrecy, patent, or copyright against free competition as to name, product, composition, or process of manufacture
3 : a business secretly owned by and run as a cover for an intelligence organization
# Definition Summary

<table>
<thead>
<tr>
<th>Term</th>
<th>Opposite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Non-Standard</td>
</tr>
<tr>
<td>Proprietary</td>
<td>Free</td>
</tr>
</tbody>
</table>
Standards - what exactly are they?

- Specification vs. Standard
- New invention or lowest common denominator
- Portability vs. Interoperability
- Based on popularity (de facto) or authority (de jure)?
- Independent Standard bodies vs. Industry consortium
Standards - why are they important?

- Limits proliferation of different ways of doing the same thing
- Establish protocols and interfaces for interoperation
- Drive mass adoption of technology
- Drives down cost of adoption
- Drives further innovation
- Gives new and small entrants the ability to leapfrog established players by avoiding vendor lock in
- Enables “best of breed” solutions
Standards - categorization

- Based on convention (Iron man Triathlon)
- Based on core vs. non-core (SOAP vs. WS-Reliable Messaging)
- Based on maturity (ANSI C, IETF EDIINT AS2)
- Based on standards bodies - W3C, OASIS, WFMC
- By vertical areas – SWIFT XML, HL7 XML
Standards - criteria for TIBCO’s participation

- Primary motivation is an area of interest
- Level of interest - is it ripe for standardization
- Level of participation
- Level of effort required
- Complexity of standards
- Weight behind a specification
Maturity Model for Specifications

- Specification
- Experimentation
- Early Adoption
- Standardization

Maturity vs. Time
### Components of an Enterprise SOA Infrastructure

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Container</td>
<td>The SOAP stack(s) that your services run on</td>
</tr>
<tr>
<td>Service Intermediaries</td>
<td>The stuff in between service providers and consumers (ESB/Fabric)</td>
</tr>
<tr>
<td>Service Registry</td>
<td>Yellow Pages/Google for you services</td>
</tr>
<tr>
<td>XML Repository</td>
<td>Where all the metadata is really stored</td>
</tr>
<tr>
<td>Web Service Security</td>
<td>Authentication, Encryption, Authorization/Entitlements</td>
</tr>
<tr>
<td>Management of Web Services (MOWS)</td>
<td>How to keep your services up and running, with SLAs and FT/Clustering</td>
</tr>
<tr>
<td>Management Using Web Services (MUWS)</td>
<td>Next generation Enterprise and Network Management (post-SNMP)</td>
</tr>
<tr>
<td>Legacy Integration</td>
<td>Connecting to what you have today Service Enablement</td>
</tr>
<tr>
<td>Orchestration and Composite Apps</td>
<td>The killer app for services is to reuse them</td>
</tr>
</tbody>
</table>
Some Services Standards to Become Familiar With

- **Data Standards**
  - **XML**
    - Standardized data structure syntax
  - **XSLT**
    - Formal schema definition for XML

- **Messaging Standards**
  - **SOAP** – Simple Object Access Protocol
    - A transport-independent protocol for peer-to-peer message exchange across a network
  - **WS-Addressing**
    - Standardized addressing and routing
    - Endpoint references extend the WSDL model

- **Service Description**
  - **WSDL** – Web Services Description Language
    - A language for describing the capabilities and requirements of the service

- **Policy**
  - **WS-Policy**
    - Simple and extensible grammar for describing policies
    - A processing model to interpret them (requires infrastructure!)
    - E.g. max message size, which headers require encryption, etc.
  - **WS-PolicyAttachment**
    - Associates policy with WSDL metadata
More Services Standards to Become Familiar With

- **Security**
  - WS-Security
    - Mechanisms to include security tokens in a message
      - Message integrity and confidentiality, single-message authentication
      - E.g. encryption key references and their association with portions of the XML
    - Provides end-to-end security
      - Removes transport from the security equation
  - WS-SecurityPolicy
    - Specifies the security requirements of a service in WS-Policy form
  - WS-Trust
    - Protocols for requesting, issuing, and brokering security tokens
  - WS-SecureConversation
    - Extensions for establishing a secure context (symmetric encryption) for a series of interactions
      - The SOAP-level equivalent of SSL
  - WS-Federation
    - Defines mechanisms for sharing identity, account, attribute, authentication, and authorization across trust domains

- **Service Discovery**
  - UDDI
    - Repository-based mechanism for sharing information about web services
  - WS-Discovery
    - Dynamic service discovery
Still More Services Standards to Become Familiar With!

- **Coordination**
  - WS-ReliableMessaging
    - Delivery semantics: at least once, at most once, in-order delivery
  - WS-Coordination
    - Framework for scenarios in which coordinators are required
  - WS-AtomicTransaction
    - 2-phase commit protocol
  - WS-BusinessActivity
    - Long-running transactions

- **Enumeration**
  - WS-Enumeration
    - Support for data exchange involving multiple interactions (i.e. arbitrary-length lists, etc.)

- **Transfer**
  - WS-Transfer
    - Basic operations for managing entities – Create, Read, Update, Delete (CRUD)
    - Resource and Factory concepts

- **Events**
  - WS-Eventing
    - Subscription management for event notification, actual notification

- **Management**
  - WS-Management
    - Operations for system management

- **Standards introduction**
Web Services Standards - maturity

- WSDM 1.0
- WS-Transactions
- WS-CAF
- WS-BPEL
- WS-Choreography
- WS-RP 1.0
- WS-RP 2.0
- WS-Security 1.1
- WS-SecureConversation
- WS-Trust
- WS-SecurityPolicy

Specifications: OASIS
Experiments: OASIS
Early Adoption: W3C
Standardization: OASIS
<table>
<thead>
<tr>
<th>Functional Area</th>
<th>Specification / Committee</th>
<th>TIBCO Leadership</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliable Messaging</td>
<td>WS-ReliableMessaging</td>
<td>• Member of WS-Reliable Exchange OASIS Technical Committee</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Co-author of spec (with Microsoft, IBM and BEA)</td>
</tr>
<tr>
<td>Events</td>
<td>WS-Eventing</td>
<td>• Co-author of spec (with Microsoft and BEA)</td>
</tr>
<tr>
<td>Alerts/Notifications</td>
<td>WS-Notifications</td>
<td>• Co-author &amp; member of OASIS Technical Committee</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Spec split into WS-BaseNotification, WS-BrokeredNotification, &amp; WS-Topics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TIBCO has driven consolidation of WS-Eventing &amp; WS-BaseNotification</td>
</tr>
<tr>
<td>Addressing</td>
<td>WS-Addressing</td>
<td>• Actively driving WS-Addressing W3C Working Group</td>
</tr>
<tr>
<td>Security</td>
<td>WS-Security</td>
<td>• Charter and voting member of OASIS Technical Committee</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Demonstrated WS-Security Interop at Gartner WS Summit April 2005</td>
</tr>
<tr>
<td>Management and Monitoring</td>
<td>WSDM (Distributed Management)</td>
<td>• Member of OASIS Technical Committee</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• WSDM spec ratified as standard and supported by TIBCO products</td>
</tr>
<tr>
<td>Orchestration</td>
<td>WS-BPEL</td>
<td>• Member of OASIS Technical Committee and key contributor</td>
</tr>
<tr>
<td>Description</td>
<td>WSDL 2.0</td>
<td>• Key W3C Working Group participant (contributed expertise in pub/sub messaging)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Pushed for inclusion of sophisticated message exchange patterns</td>
</tr>
<tr>
<td>Transport</td>
<td>SOAP 1.2</td>
<td>• Key W3C Working Group participant</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Have obtained 100% interoperability</td>
</tr>
<tr>
<td>Pluggable Service Engines (Open ESB)</td>
<td>Java Business Integration (JSR 208)</td>
<td>• Member of JCP Expert Group</td>
</tr>
<tr>
<td>Transactions</td>
<td>WS-TX (Transactions)</td>
<td>• Member of OASIS Technical Committee</td>
</tr>
<tr>
<td>Security</td>
<td>WS-SX (SecureExchange)</td>
<td>• Member of OASIS Technical Committee</td>
</tr>
</tbody>
</table>
Web Services - WS-BPEL

- **What is WS-BPEL?**
  - Defines a syntax for the orchestration of existing Web Services. Dependent on WSDL, XML Schema, XPath, etc.

- **Features**
  - Ability to combine block-structured and graph-structured paradigms
  - Ability to specify compensation of faulted scopes
  - Event handling
  - Late Binding

- **Application**
  - B2B Public Processes
  - Protocol definition with some visibility into each side’s state

- **Status**
  - OASIS Standard expected October 2006
Workflow - maturity

BPEL
BPMN 1.0
XPDL 2.0

Specification  Experimentation  Early Adoption  Standardization
Web Services - WS-Reliable Messaging

- **What is WS-ReliableMessaging?**
  - A messaging protocol that allows service producers and consumers to reliably communicate in the presence of component, system and network failures.

- **Features**
  - Reliable messaging endpoints responsible for providing delivery assurance
  - Support for At Most Once, At Least Once and Exactly Once and In Order delivery assurance
  - Transport independent protocol definition
  - Defines SOAP bindings for this protocol
  - Support for both ACK and NAK based protocols
  - Endpoint capability specified using mechanisms specified in WS-Policy and WS-Policy Attachment

- **Application**
  - End to end reliable delivery of messages over unreliable transport

- **Status**
  - OASIS Standard expected Q3 2006
Messaging - maturity

JMS 1.1

WS-ReliableMessaging

Specification  Experimentation  Early Adoption  Standardization
What is JBI?
- Defines a Service Oriented Architecture based infrastructure for integration.

Features
- Notion of Binding Component (BC) and Service Engine (SE) - generically referred to as JBI component
- Support for pluggable components from different vendors
- Concept of a Normalized Message Router that decouples BCs and SEs
- Defines packaging of JBI components and deployment of service artifacts to them
- Management interfaces for managing the JBI environment

Application
- Build Service Engine, eg. BPEL engine
- Build Binding Component, eg. SOAP/HTTP

Status
- Final Release August 2005
ESB - maturity

Not even going to try ;-)
Standards support in TIBCO products (current/future)

- SOAP 1.1 and 1.2 – BusinessWorks, Matrix
- WSDL 1.1 – BusinessWorks, Matrix
- WS-Addressing – BusinessWorks, Matrix
- WS-BPEL - BusinessWorks
- WSDM - EM Advisor
- WS-ReliableMessaging – BusinessWorks, Matrix
- WS-Security – BusinessWorks, Matrix
- WS-Eventing/Notification – BusinessWorks
- UDDI v1 v2 v3 – BusinessWorks, PortalBuilder
- JBI – Matrix
- JMS - EMS
- WSRP - PortalBuilder
What are we leading?

- Standardization of Service Container – JBI (JSR208)
- Service Orchestration – WS-BPEL
- Business Process Management – XPDL, BPMN
- Reliable Messaging – WS-ReliableMessaging
- WSDL 2.0 – Support for Publish/Subscribe MEPs
- Eventing and Notification – WS-Eventing and WS-Notification
- SOAP over JMS Standardization
What are we learning?

- Standardization process is messy
- Composition of various WS standards is a challenge
- Just because a standard is out of the gate first it may not win
- Scaling is still a question mark
- Interoperability
- Performance issues
- Pragmatic approach
SOA Best Practices
Seven SOA Best Practices

1. Ensure every project has a positive ROI
2. Establish effective service governance
3. Organize governance around your organization
4. Assign the 5 key leadership roles for every project
5. Go beyond request-reply
6. Use standards where applicable, appropriate, and mature
7. Build your SOA to support all your technologies and platforms
1. Ensure every project has a positive ROI!
The “Big” ROI: End-to-End Process Improvement
Case Study: Con-Way Transportation Services

- Con-Way is a subsidiary of CNF Inc. – a $3.7 billion publicly traded company on the New York Stock Exchange (Ticker Symbol: CNF)

- Con-Way Transportation Services offers time-definite and day-definite freight transportation for commercial, industrial, retail, wholesale and manufacturing companies throughout N. America
  - 19,431 Employees
  - More than 445 Service Locations
  - 28,061 Trucks, Tractors and Trailers
  - $2.6 Billion Gross Revenue
TIBCO is Helping Con-Way Deliver Real Value with SOA

- Shortened time-to-market by over 40% for new major business initiatives through reuse of existing services
- Reduced development costs by writing less code
- Event-driven integration with US/Canadian Customs processing has reduced border crossings from 2-3 hours to less than a minute
- Eliminated redundant data entry and manual processes saving up to 500 man-hours daily
- Zero latency in the flow of mission critical data providing real-time decision support for Operations, Sales and Finance personnel
- SOA facilitating change in business and technical areas by providing integration hooks for advanced technologies and e-Business processes such as dock simulation, dock automation, handheld devices, etc.
- Con-Way has won several awards - CIO-100 award ’04, CIO-100 award ‘05 and InfoWorld 100 Innovation ’05 in recognition of their SOA/EDA implementation
Con-Way Recommendations for Successful SOA

- Recruit Sponsorship from IT Executive Management

- Establish Services Governance:
  - Educate developers to first look for existing services before building their own
  - Gain developer trust in services they do not control by ensuring good quality
  - Choose right component boundaries and service granularity to foster reuse
  - Create meaningful repository of services and events across technologies

- Develop canonical model of key business objects within your enterprise to be used in the integration layer

- Architect for adaptability to ensure new technology approaches can be adopted without major impact to existing implementations

- Embrace industry standards where possible to ensure software vendor and hardware platform neutrality
• World Third Largest Gaming Company
• 28+ Casinos in 13 States in US
• “A Technology-enabled Marketing Company”

TIBCO Projects Since 2001
• Total Rewards Loyalty Program
• Integration with 300GB+ Teradata, 45000+ Devices, Gaming, Hospitality and CRM Systems
• Employee Portal for 42000+ Employees
Old Architecture

Technical Challenges
• Data is distributed and duplicated
• Lack of data consistency
• Hard-coded, customized connections
• Cannot achieve real time CRM because of batch environment
• Unable to make real time offers based on guest preferences
Harrah’s New Enterprise Architecture - 2005

Move to SOA
- Breaking away from monolithic applications
- Minimizing point to point interactions
- Orchestrating business process based on business event
- Creating reusable services (logic and data is duplicated across several applications such as Customer Preference)
- Standard way to interface with external partners
Some Example Benefits – Harrah’s

- **TIBCO ESB (BusinessWorks) helped deliver SOA**
  - Reusable services reducing development time by 60%

- **Improved Customer Service through Wireless Handheld Access**
  - Guest services can look up patron info while on property floor
  - CSRs can interact with customer using real-time data

- **Cost-effective Regulatory Compliance: Responsible Gaming II (RG2) Initiative**
  - Business rules dictate how patrons should be handled (e.g., when to provide credit, check cashing services, direct marketing, or exclusion from casino and gaming)

- **The Long-Term Payback: Acquisition of Caesar’s**
  - All newly acquired properties can view Harrah’s patrons through single UI by integrating 23 disparate systems
    - 3 years earlier integrating 5 casinos took 2 years
    - Caesar’s 18 casinos were integrated in 6 months
ROI and Services

- Achieving the “Big” ROI requires a sustained investment in services.

- Sustaining the investment requires shorter-term payback to cover costs:
  - Every individual project needs its own positive ROI based on the business value provided by the project.
  - Service development costs can be absorbed by accepting a smaller (though still positive) ROI on projects.

- Letting individual project ROI cover the service development costs allows service investments to be sustained indefinitely.

- This formula for success is proven:
  - Con-Way Transportation Services
  - Harrah’s Entertainment
Keep Focused on the ROI – Not the Technology!

- Services are not about technology!

- Services are about ROI – and actually achieving it

- Focus should be on reusable functionality
  - Identifying what is needed
  - Making sure it is leveraged

- Technology issues are secondary!
2. Establish effective service governance
Governance Questions

- **Service Creation and Utilization in Projects**
  - How/when will potential services be identified?
  - How/when will we ensure that existing services are used where appropriate?

- **Service Evaluation and Specification**
  - How will services steering committees be organized?
  - How will they get engaged with projects?
  - How will this interaction be managed?

- **Service Operation**
  - How will the operation of the service be coordinated with dependent applications/services?
  - How will changes to services and the corresponding changes to dependent applications/services be managed?
    - How will we get that last user off the old version of the service?
Services Governance

- Registry
- Repository
- Security
- Policy
- Services Management

Enterprise Service Bus (ESB)/Backbone

Registry and Repository

Centralized Policy & Services Management

Place Order
Check Customer Account
Check Quantity
Credit Check
Alert Large Order
Process Order
Notify Customer
Issue Invoice
3. Organize governance around your organization
Services Steering Committee Organization

- **Centralized approach**
  - One organization evaluates all service proposals and specifies services
  - Limited in scalability – potential bottleneck
  - Appropriate for small-to-mid-size companies with centralized development organizations
    - e.g. Harrah’s

- **Siloed Approach**
  - Each major silo (application area) has a steering committee consisting of
    - Representatives of the silo itself
    - Representatives of the silos that need to interact with it
    - Enterprise architects (business process and systems)
  - Appropriate for large companies with distributed development
    - PepsiCo, FedEx, Citigroup

- **Remember- a process must be in place to:**
  - Engage the steering committee
  - Determine who will implement the service
  - Determine who will absorb the additional costs (there WILL be extra cost!)
4. Assign the 5 key leadership roles for every project
Five Key Roles drive end-to-end business process improvement, which drives business ROI:

- **Project manager**
  - Broader responsibility to deliver business value, not manage development. Projects must be organized with this end-to-end focus

- **Business process architect**
  - To achieve business ROI, an architect should be focused on the design of the business process

- **Systems/software/enterprise architect**
  - Technologies supporting the business process must have an end-to-end focus

- **IT executive sponsor**
  - Identify IT resources and resolve organizational issues

- **Business executive sponsor**
  - Establish the goals, explain the value, drive business commitment and ensure funding
  - The beneficiary of the ROI!
5. Go beyond request-reply
There are three styles of services:

- **On-Demand (request-reply)**
  - Service waits for requestor to invoke an interface and then initiates the requested action
  - E.g. “place an order” at PepsiCo

- **Event-Driven**
  - Upon receipt of an event, the local service performs its required function
  - Service proactively notifies subscribers when specific events occur
  - E.g. “card-in” and “card-out” at Harrah’s

- **Continuous**
  - Service that runs on its own without formal invocation either periodically or continuously
  - E.g. process monitoring services such as TIBCO Business Factor
Think Beyond Request-Reply

- When you order a book at Amazon.com, the book does not arrive while you are at the keyboard
  - The order is acknowledged via request-reply
  - The processing of the order occurs asynchronously
  - The “service” of order management encompasses both

Real-world business processes require more than request-reply services!
6. Use standards where applicable, appropriate, and mature
Use Standards where Applicable, Appropriate and Mature

- **You would be hard-pressed to develop a standard of equivalent quality**
  - Avoid rolling your own (re-inventing an existing wheel)

- **Recognize that standards are not yet mature**
  - e.g. WS-Eventing vs. WS-Notification
  - Modular WS- standards ease the evolution

- **Plan for standards evolution**
  - E.g. Adopt SOAP-Envelope now for message data structures
  - Migrate to full SOAP services with policy and security support as the standards mature and the supporting infrastructure becomes available
The Standards Don’t Cover it All!

- Many standards require infrastructure investments
  - WS-Security, WS-Policy require a credentialing infrastructure

- UDDI standardizes the mechanics of accessing information about services, but does not standardize the content
  - Your own policies and practices must manage the content

- WSDL will not tell you the design intent of the service
  - When you should or should not use the service
  - Additional documentation is required!

- You probably want to control and manage the actual access to services
  - Both for capacity planning and for access control purposes
  - You need processes for this
7. Build your SOA to support all your technologies and platforms
Don’t Tie Services to an Application Server

<table>
<thead>
<tr>
<th>App Server Architecture</th>
<th>SOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homogenous</td>
<td>Heterogeneous</td>
</tr>
<tr>
<td>Language dependent</td>
<td>Language independent</td>
</tr>
<tr>
<td>Centralized application tiers</td>
<td>Massively distributed services</td>
</tr>
<tr>
<td>Code centric applications</td>
<td>Flexible composite applications</td>
</tr>
<tr>
<td>Request/reply driven</td>
<td>Request/Reply, Pub/Sub, …</td>
</tr>
</tbody>
</table>
Build a Heterogeneous, Distributed SOA

Service Governance and Assurance

- Security
- Registry
- Policy Management
- Service Management
- Transaction Coordinator
PepsiCo Enterprise Services Architecture

Managed Business Processes (BW + BPM)

Common Services Layer (BW + Core Technologies)

Business Services
- Order Validation
- Product Pricing
- Shipping Distance Calculation
- Tax Lookup

Infrastructure Services
- Security Provisioning
- Error Handling
- Cross-Referencing
- Common Logging

Data Services
- Master Data Services
- Data Cleansing
- Metadata Services
- Enterprise Data Model

Process Integration (TIBCO BW)

Enterprise Service Bus (TIBCO EMS for internal services, BW as an ESB Gateway)

Data Access
- EDW
- SAP-BW
- ERP (SAP)
- Data Access
- Data Access
- Data Access
- Data Access
- Data Access

ETL (File based and Database table oriented high volume data)

© 2006 TIBCO Software Inc. All Rights Reserved. Confidential and Proprietary.
Summary
Questions?