IBM SOA Architect Summit

SOA on your terms and our expertise
Application Architecture: Reusing Existing Applications in SOA-Based Business Processes

Braden Flowers
Worldwide WebSphere Integration Solution Architect
SOA Architect Summit Roadmap

What is the impact of SOA on current Enterprise Architectures?
- Alignment of Business and IT Architectures
- SOA Reference Models
- SOA Governance

How do you develop SOA with a business focus?
- Business Components
- SOA Design
- Business Process Management

How do you reuse applications in the context of SOA?
- Asset Discovery
- Application Reuse

How do you leverage information in an SOA?
- Information as a Service
- Master Data Management

How does my infrastructure support SOA?
- Service Management / QoS
- Security

Alignment of Business and IT Architectures

IBM SOA Architect Summit
SOA and Application Architecture

- SOA Application Architecture Considerations
- SOA Application Architecture Best Practices
- IBM Capabilities to Support SOA Application Architecture
- Summary
SOA Enables Greater Reuse of Existing Assets

= Assets for potential reuse

- getCustomerAddress
- updateCustomerAddress
- createAccount
- conflictCheck
- adjustInternalRating
- initializeAccount
- verifyCreditRating

SOA on your terms and our expertise
Application Architecture Considerations

- Analyze business processes to discover services
  - Identify services required to perform the individual tasks defined by a given business process
  - Analyze existing applications to identify service providers

- Creating services
  - Use best practices (defined as patterns) to create services from existing assets
    - Use tools and standards to service enable an asset
    - Use externally provide services to support commodity tasks
    - Fill in gaps by creating new services

- Connecting to service providers
  - Enable "any-to-any" linkage and communication between services inside and beyond the enterprise
  - Simplify connectivity by providing infrastructure that ensures Qualities of Service (QoS) including security, reliability, and scalability
SOA and Application Architecture

- SOA Application Architecture Considerations

- SOA Application Architecture Best Practices
  - Asset Discovery Approaches
  - Reuse Patterns
  - Service Connectivity Scenarios

- IBM Capabilities to Support SOA Application Architecture

- Summary
Bottom Up SOA Approach

Legacy application

Business Requirements

Business use Cases

Business Analyst

Interface Specification

WSDL

Services (reusable functional components)

New Application

Choreography (business rules and processes)

IT

Business Analyst

IT

Business Analyst

IT

Business Analyst

Business Analyst
Existing Asset Analysis

Coarse-Grained Mapping of Candidate Services to Existing Applications

- Examine assets (existing custom, packaged applications, and industry models, etc.) to determine what can be leveraged to realize service functionality.
- Understand the business functions supported by each application.
- Record attributes of existing applications in terms of technologies used, architectural styles, and so on.
- Identify applications that perform common services.
Identifying Services and Service Components

From Asset Analysis
Asset Analysis and Transformation Methods

Asset Analysis

- Deep interactive analysis
- Find and manage business rules
- Create components from existing code
- Analyze code for SOA reuse

Asset Transformation

- Application discovery
- Application analysis
- Discover relationships between applications
Supporting Services Through The Lifecycle

**Service Development**
- Service Discovery
- Service Development Lifecycle

**Service Deployment**
- Runtime Repository
- Runtime Discovery

**Development Registries**
- Version Control
- Asset Development Management

**Service Asset Manager**

**Service Registry & Repository**

**Other Service Endpoint Registries / Repositories**
- UDDI Registries
- Info-based Services
- Other External Reg / Rep

**Service Management**
- Operational Efficiency & Resilience
- Configuration Data Discovery
- Managing change

---

IBM SOA Architect Summit

SOA on your terms and our expertise
Service-enable Mainframe Assets

Make Better Use of CICS & IMS Investments

Enterprise Service Bus
- Messaging provides loosely coupled connectivity with assured delivery and reliability
- Advanced ESB Solutions can convert from any format (including SOAP) to COMMAREA format
- No changes required to existing application

Adapters
- J2EE to Mainframe adapters provide tightly coupled connectivity with two-phase commit support

Native Web Services
- CICS & IMS can both expose transactions as native Web Services
- No other runtimes required

Adapters Diagram:
- J2EE to Mainframe adapters
- WebSphere for z/OS + CICS TG
- CICS Applications
- Intranet/Extranet/Internet

SOA on your terms and our expertise
Service-enable Packaged Applications
Make Better Use of Commercial Software and Other Technology Investments

- **Service enablement** – provide a standard interface to proprietary implementations
- **Service discovery** – browse, select, and generate service descriptions from the application repository
- **Event capture** – detect and publish application events and control in-bound / out-bound information flows
- **Pre-built or build your own** – vast range of ‘ready-to-go’ adapters and toolkit to generate your own
- **Enterprise Quality of Service** – ensure mission-critical quality of service
Service Enable “Component” Applications

**Partner or Customer (External)**

*Consume services*

- **J2EE application**
  - Java client
  - Web Services client
  - `<consumer>`

- **.NET client**
  - .NET
  - Web Services client
  - `<consumer>`

- **Partner Service Provider**
  - Business Logic
  - Web Services
  - `<provider>`

**Enterprise (Internal)**

*Exterrnally Expose services*

- **Web Services Provider**
- **Services Gateway**
  - SOAP / HTTP
  - SOAP / HTTP(s)
  - WS Security

- **J2EE application**
  - Java client
  - Web Services client
  - `<consumer>`

- **J2EE application**
  - Java client
  - Web Services client
  - `<consumer>`

**Consume services**

- **SOAP / HTTP**
- **SOAP / HTTP(s)**
- **WS Security**

- **SOAP / JMS**
SOA and Application Architecture

- SOA Application Architecture Considerations

- SOA Application Architecture Best Practices
  - Asset Discovery Approaches
  - Reuse Patterns
  - Service Connectivity Scenarios

- IBM Capabilities to Support SOA Application Architecture

- Summary
Direct Access Pattern

Benefits:
- Shorter deployment cycle … compared to indirect access
- The service interface is defined by the asset
  - No analysis required to determine the interface
- No knowledge of other runtimes (Java, Message Broker, etc.) is necessary
- Fewer platforms/moving parts

Issues:
- Consumers become coupled to the asset environment
  - Difficult to substitute the asset for an alternate
- Requires the asset runtime environment have support for service invocation
- Asset capability needs to match the service requirements
- Places an XML processing burden on the asset runtime
  - Systems that are often paid for on a “MIPS consumed” model
Direct Access Pattern Example
*CICS and IMS Native Web Services*

**CICS Transaction Server**
- SOAP
- HTTP
- WebSphere MQ
- SOAP body
- XML
- Data Mapping
- Language structure 0101001
- Pipeline

**Information Management System**
- IMS
- IMS Connect
- OTMA

**Business Logic**
- IMS
- IMS Connect
- Business Logic

**SOA on your terms and our expertise**
Direct Access Pattern Example

Java and .Net Applications
Indirect Access Pattern

Benefits:

- Business alignment is maintained
  - Service interface that suits/aligns with the business view and not with existing legacy assets
  - Service component maps between the two worlds
- Straightforward to substitute the asset
  - Service component may be replaced without impact on the consumer
- Offloads the XML processing burden
  - Many systems account for resources using a “MIPS consumed” model
- A service may be implemented using behavior from more than one asset
  - Service component aggregates the behavior to realize the service
  - Enables additional capability to be added

Issues:

- Longer deployment cycle than Direct Access
  - Consideration must be given to the definition of the service interface
  - Time spent developing the service component
- More complex than Direct Access
  - Generally involves the use of connector/adapter technology between the service component and the backend systems
  - Usually introduces a middle tier
Indirect Pattern...

A service component serves as a front-end providing indirect access to the target asset

- Sub Pattern: Adapter, Gateway
  - Proxy based access to asset
  - Generally used to map standards based interface to asset based interface

- Sub Pattern: Application Server
  - Generalized capability for interacting with multiple target assets
  - Provides an environment for augmenting an asset’s capabilities

- Sub Pattern: Enterprise Service Bus
  - Generalized capability for interacting with multiple target assets
  - Provides an environment for augmenting an asset’s capabilities

- Sub Pattern: Terminal “Emulation” Environments
  - Encapsulating a sequence of screen interactions as a “macro”
  - Exposing a “macro” as a SOAP-based service
Indirect Pattern Example

Application Server / Adapter / Gateway

WebSphere Application Server

JCA Adapter For Siebel
JCA Adapter For SAP
IMS Connector For Java
JCA Adapter for CICS
EJB client

IMS Connect
CICS Transaction Gateway
Java ORB
RMI
IIOP

CICS Transaction Server

Information Management System

SAP
Business Logic
Business Logic
Business Logic
SAP

Indirect Pattern Example
Application Server / Adapter / Gateway
Indirect Pattern Example

**Enterprise Service Bus**

- **WebSphere Message Broker**
- **Oracle**
- **CICS Transaction Server**
- **Information Management System**
  - IMS/MQ Bridge
  - Business Logic
- **CICS/MQ Bridge**
- **Business Logic**

SOA on your terms and our expertise
Indirect Pattern Example

Terminal Emulation

WebSphere Application Server

Host Access Transformation Services

Information Management System

CICS Transaction Server

System z

System i

Indirect Pattern Example

Terminal Emulation
Combinations of Direct and Indirect Patterns

CICS Service Flow Modeler

CICS Transaction Server

Rational Developer for System z

SOA on your terms and our expertise
# Patterns Selection Guide

*Comparison of Indirect vs Direct Access*

<table>
<thead>
<tr>
<th>Decision Criteria</th>
<th>Indirect</th>
<th>Direct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementing an existing or business driven service definition</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Many requestors or requestors outside of providers domain</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Aggregation or business logic applied across multiple existing functions</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Need to enable service provider/implementation replacement</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Return subset of information available in the existing function</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Cost of MIPS on existing platform is key concern</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Several assets to be aggregated together</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Skills only available on existing platform</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Existing platform is strategic platform</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Expediency is key driver</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
Reuse Pattern Architectural Decisions

- Common deployment decisions
  - If indirect pattern, location of proxy component relative to consumer and provider
  - Placement of service boundary
- Transaction management
  - Handling rollback and compensation
- Identity management
  - Mapping to the security requirements of the asset
- End to end management
  - Providing visibility across composite application
- Likelihood of changes to the service interface
- Chargeback for use of an asset
SOA and Application Architecture

- SOA Application Architecture Considerations

- SOA Application Architecture Best Practices
  - Asset Discovery Approaches
  - Reuse Patterns
  - Service Connectivity Scenarios

- IBM Capabilities to Support SOA Application Architecture

- Summary
Core Principles of the ESB Architectural Pattern

- ESB inter-connects requestor and provider
  - Handle multiple communication protocols supporting interaction patterns
  - Flexibility to support message content models based on meta models
  - Enable interactions through defined mediation flows to process request messages and correlated results using defined patterns

- ESB provides **Service Virtualization** of:
  - *Identity* via routing
  - *Protocol* via conversion
  - *Interface* via transformation

- ESB also enables **Aspect Oriented Connectivity**
  - To handle security, management, logging, auditing, etc.
ESB Mediation Base Patterns

- Aspect oriented connectivity
  - Monitoring, logging and access to the services registry
  - Security and management policy definition points

- Service virtualization
  - Route messages dynamically based on defined service metadata derived from registry
  - Handle differing QoS for services

- Compositions
  - Complex event processing, e.g. handling failover
  - Logging the event and processing results
ESB Gateway Pattern

- Variant of routing or protocol switch pattern which maps service endpoints, possibly providing security functions (authorization and access control) and logging or auditing capabilities.
- May incorporate transform and monitor mediations to provide encryption and logging, or auditing. It may also aggregate and disaggregate messages in a one-to-many relationship.
- Example: Service portals which act as a single point of contact for multiple services and hide the details of “internal” services.

Key context Issues:
- Security
- Quality of Service
- Management
- Transactions
ESB Integration Topology Patterns

Direct
- Multiple namespaces and administration domains
- Namespace mapping in each ESB
- Services are likely to be applicable throughout the enterprise

Brokered
- Multiple namespaces and administration domains
- Namespace mapping in gateway facilitates service interaction
- Subset of services applicable throughout the enterprise

Federated
- Multiple namespaces and administration domains
- Namespace mapping in Federated ESB facilitates service interaction with multiple implementations
- Subset of services applicable throughout the enterprise
SOA and Application Architecture

- SOA Application Architecture Considerations
- SOA Application Architecture Best Practices
- IBM Capabilities to Support SOA Application Architecture
- Summary
Mapping to the IBM Products

- Rational Developer for System z
- WebSphere Development Studio Client for System i
- WebSphere Studio Asset Analyzer
- Rational Transformation Workbench
- WebSphere ESB
  - WebSphere Message Broker
  - WebSphere DataPower Integration Appliance
  - WebSphere Service Registry & Repository
- WebSphere Partner Gateway
- WebSphere Application Server
- WebSphere Adapters
  - CICS Transaction Gateway
  - WebSphere Host Access Transformation Services
- WebSphere Message Broker
- WebSphere DataPower Integration Appliance
- WebSphere Service Registry & Repository
- WebSphere Application Server
- WebSphere Adapters
  - CICS Transaction Gateway
  - WebSphere Host Access Transformation Services

Business Services
- Supports enterprise business process and goals through businesses functional service

Process Services
- Orchestrate and automate business processes

Information Services
- Manages diverse data and content in a unified manner

Interaction Services
- Enables collaboration between people, processes & information

Partner Services
- Connect with trading partners

Business App Services
- Build on a robust, scalable, and secure services environment

Access Services
- Facilitate interactions with existing information and application assets

Infrastructure Services
- Optimize throughput, availability and utilization

Management Services
- Manage and secure services, applications & resources
Summary

- There is significant value in reusing existing assets
  - Faster time to value
  - Cheaper to re-use than to re-write
  - Existing assets are tried and trusted

- Well defined approaches to discovering high-value assets for reuse
  - Analysis done as part of service design methodology (e.g. SOMA)
  - Existing asset analysis through tools

- Two primary architecture patterns for reusing existing applications
  - Indirect access to target asset through service component
  - Direct access to target asset through service interface

- Need capabilities to support connecting and using existing assets:
  - Enterprise Service Bus provides main capabilities to connect and use existing assets
  - Service registries and repositories to support service through lifecycle
SOA on your terms and our expertise