The Art & Science of Service Design

Jason Bloomberg & Ron Schmelzer
ZapThink LLC

SOA'09
SOASUMMIT2009

MAY 2009 | SCOTTSDALE, AZ

What is SOA?

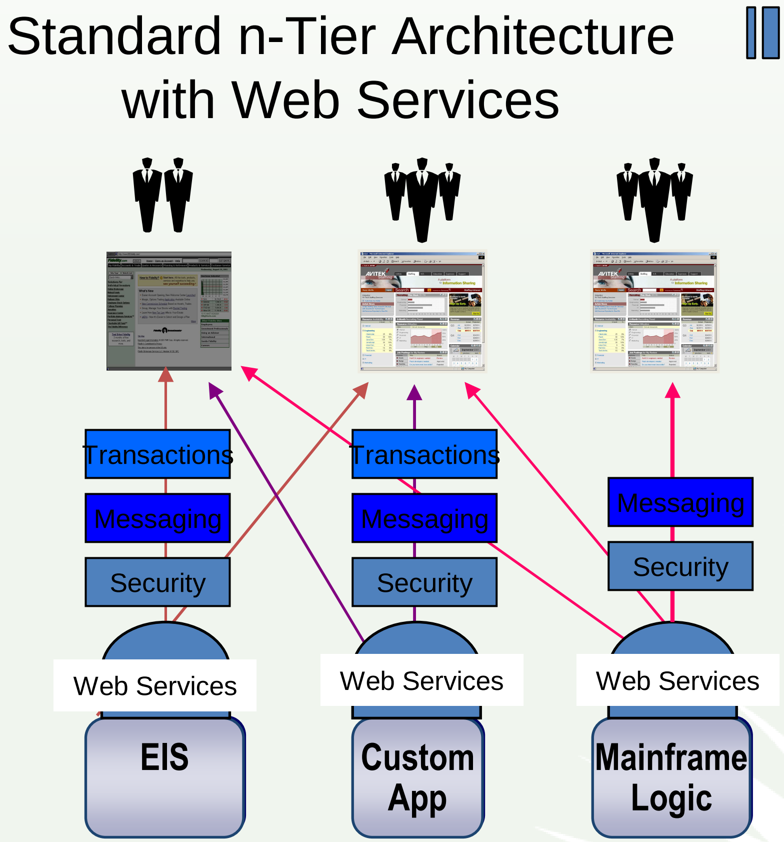
- SOA is *architecture* – a set of best practices for the organization and use of IT, and the *discipline* to follow them
- Abstracts software functionality as loosely-coupled, Business *Services*
- Services can be composed into applications which implement *business processes* in a flexible way, without programming

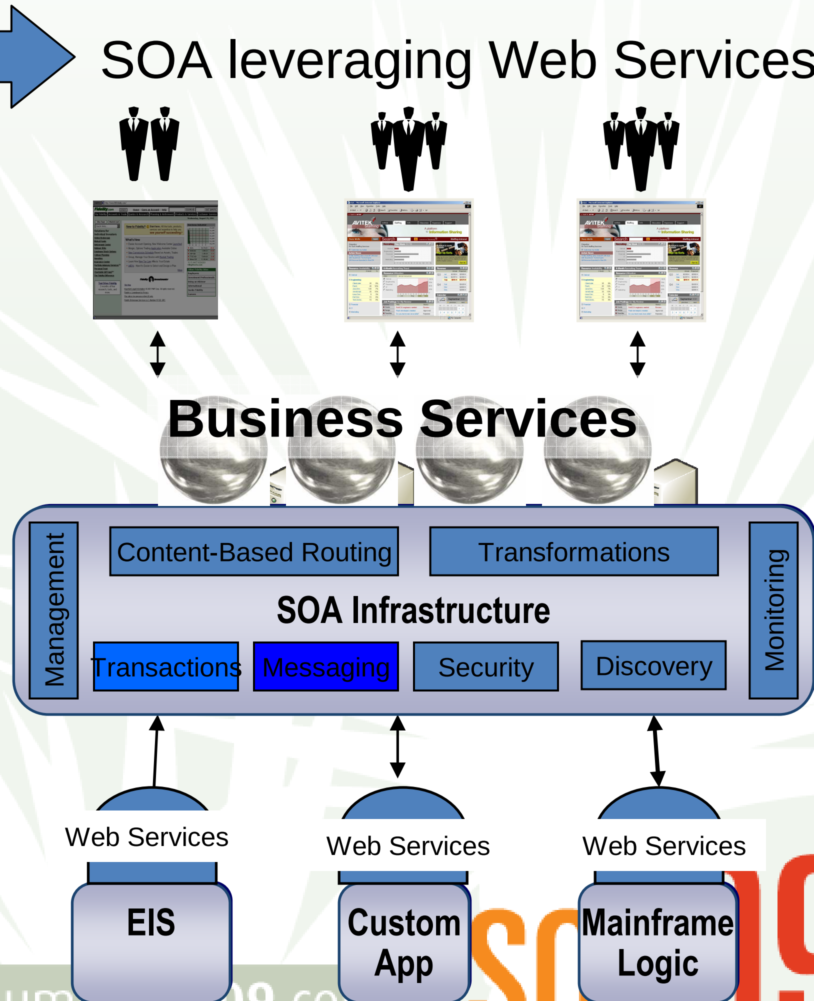
Lego "Ilities"

- Interoperability
- Unbreakability
- Composability
- Reusability

What the Business wants from IT!

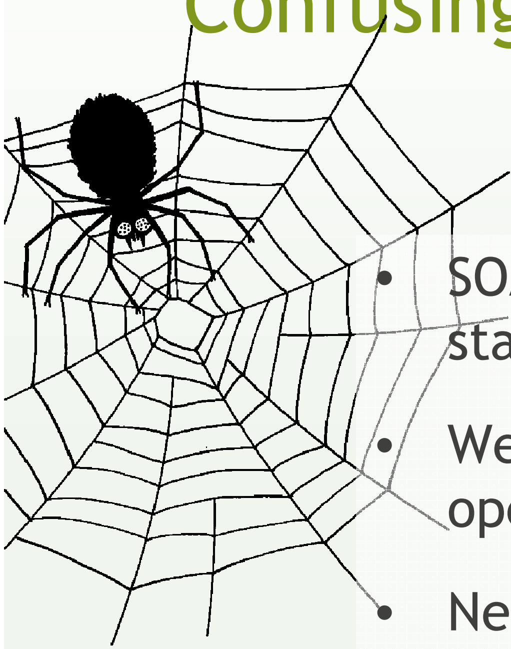
SOA vs. Web Services





Source: HP, ZapThink

Confusing SOA & Web Services



- SOA is an architectural approach, Web Services are standards-based interfaces
- Web Services serve a role in SOA implementations when open standards are required
- Neither is necessary for the other
- Vendors promote this confusion by calling Web Services integration “SOA”

If not Web Services, Then What?

- To be a Web Service, a Service interface *must* conform to WSDL
- WSDL is seriously lacking in many respects
- So, many Services follow contract templates that go beyond WSDL
- The architects’ dilemma: Web Service or Service with internal standard contract template?

What's a Service?

- Word has many meanings
- Even in IT, has many meanings
  - Software-as-a-Service
  - IT Service Management
  - SOA

Levels of Service Abstraction

- Service Implementation
  - Working software that implements the Service

- Service Interface
  - Contracted interface to underlying functionality (includes Web Services)

- Business Service
  - Abstraction of underlying functionality and data with a clear business context

Consumers & Providers

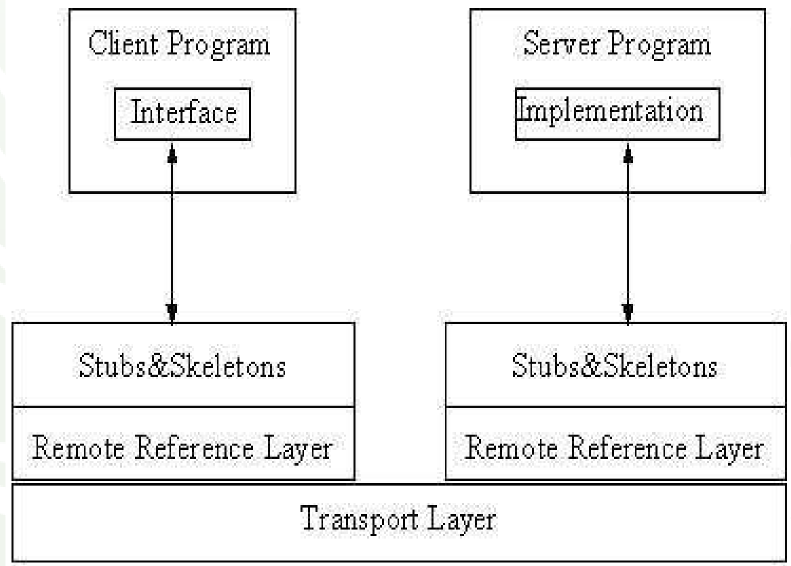
- Service Provider – software endpoint that provides a Service interface
- Service Consumer – software endpoint that interacts with a Service Provider as per the Service Contract
- *Providers* and *consumers* are patterns – the same software might be both at once
- Providers and consumers should generally be *loosely coupled* from one another

What are Services?
(Technically, Service *interfaces*)

- Contracted interfaces to software functionality and data that communicate via messages

Interoperability vs. Portability

- Traditional RPC-based distributed computing based upon *portability*
  - Object serialization
  - Remote method invocation
  - Simulate one local computer



- SOA based upon *interoperability*
  - Code remains in place, known only by contracts
  - Computers interact via messages

"Write once, run anywhere" gives way to "write once, <u>*access*</u> anywhere"

Source: http://www.cc.gatech.edu/~yongzhu/Cs7001/cs7001_2_report.html

Service Interfaces Aren't Good Enough!

- Service Interfaces provide a *technical* context
  - Standard interfaces like Web Services help, but don't provide all the benefits of SOA

- Business Services provide the *business* context
  - The business doesn't care *how* it works, as long as it does!

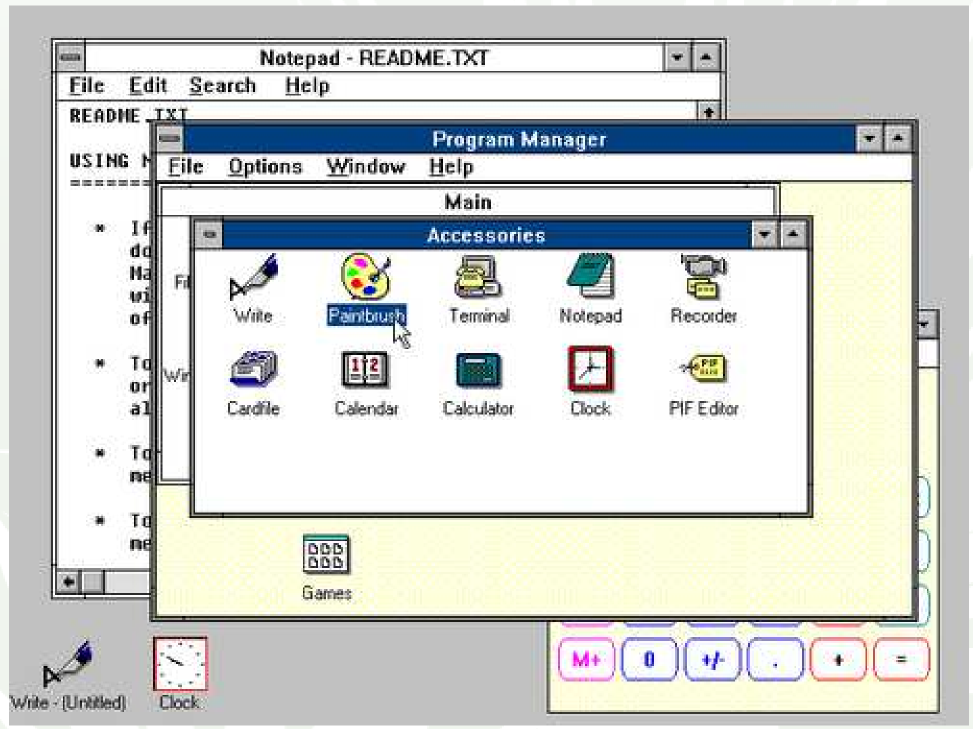
The Difference is the Abstraction

- Mature technology is complex on the inside yet simple on the outside

- The secret is the *abstraction layer*

software AG

Abstraction = *Working Illusion*



software AG

Building a Working Illusion

- Service abstraction provides simplicity to user
- Requires additional complexity "behind the scenes"
- Sleight of hand & misdirection!

The Fundamental Technical Challenge of SOA

Creating & Maintaining the Service Abstraction

- Location-independent, loosely coupled *business Services* that:
  - Provide business agility
  - Support flexible business processes
  - Empower users
- At a higher level of abstraction than a Service interface

Sounds good, but how does it work?

Multiple Interfaces per Implementation

One Service implementation that supports multiple consumers

Examples: Multiple Interfaces Per Implementation

- Supporting consumers with different form factors
  - Examples: Mobile, desktop, browser
- Supporting users with different data format requirements
  - Examples: Differing address formats, schema versions
- Handling Service versioning
  - Each interface a different "Service version"

Multiple Implementations per Interface

May have multiple contracts

Examples: Multiple Implementations per Interface

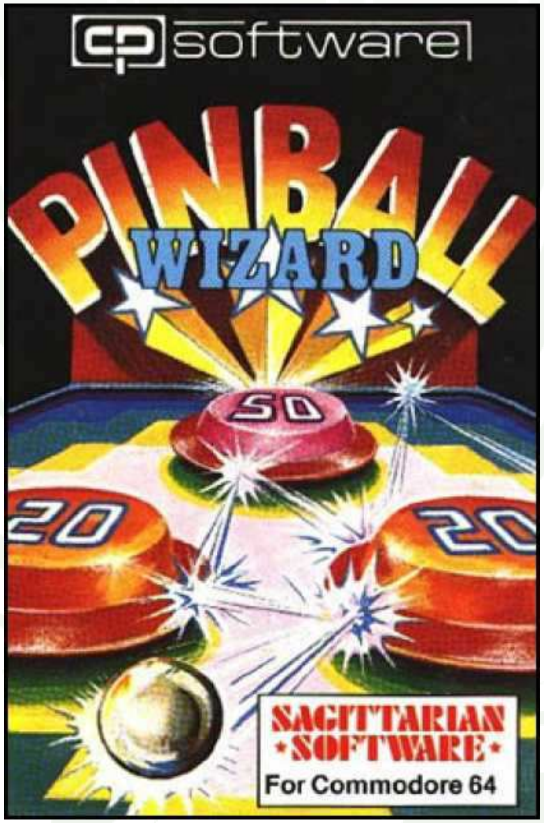
- Load balancing/failover
  - Transparent at the interface level (single contract)
- Service levels
  - Example: Bronze, silver, gold customers (multiple contracts)
- Location independence of infrastructure
  - Implementations can be local or remote as appropriate (single contract)

Multiple Interfaces per Business Service

Provides the business abstraction

Actualizing the Business Service Abstraction

- The business sees a single, location independent Business Service
- Requests to Business Service routed/transformed as necessary as a matter of policy
- Business Services abstract both data and application functionality/transactionality



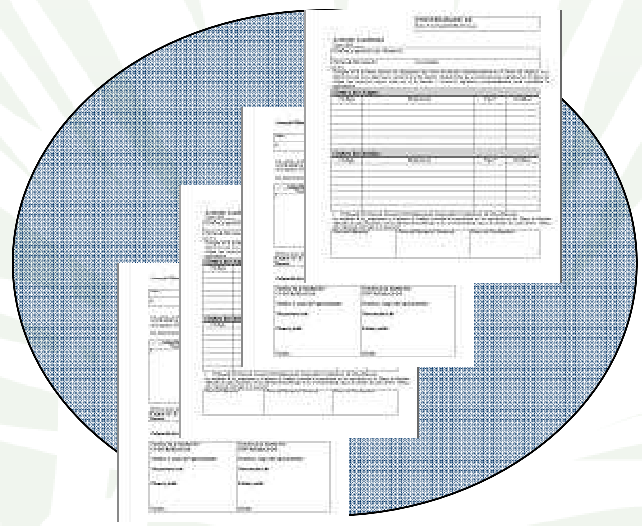
The Secret Sauce: Metadata

- Enables composite applications to be built *declaratively* (through configuration), not programmatically (with code)
- *Business* logic in composite applications represented in metadata
- Puts business logic in the hands of business users!

software AG

Metadata for SOA

- Contract metadata
  - What a Service should do (functional)
  - How a Service should behave (non-functional)
- Process metadata
  - How processes are configured
  - How processes are composed
- Policy metadata
  - What rules apply in various situations
- Schemas
- Others…



What's a Service Contract?

- *Service contract* – a document external to each participant that provides the information each participant needs to interact with the other

- *Web Services Description Language* (WSDL) – standard Service contract language
  - *Woefully inadequate!*

What’s in a Contract?

Functional Requirements

- Service Description, Service Operations
- Infrastructure (Port Type, Binding)
- URIs (Schemas, Namespaces)

Non-Functional Requirements

- Security
- Transactionality
- Consumer Requirements
- Commerce / Biz Rules
- Semantics
- Quality of Service Levels
- Process Instance State
- Message Exchange Patterns
- Testing
- Versioning
- Deployment Details

What's NOT in the Contract

A contract is an expression of
<u>visible</u> aspects of Service behavior.

It does NOT specify:

- Service Implementation Details
  - Programming model
  - Object references
  - In-memory representations
- Examples
  - Exposing Java Classes
  - Serialization of Java Objects

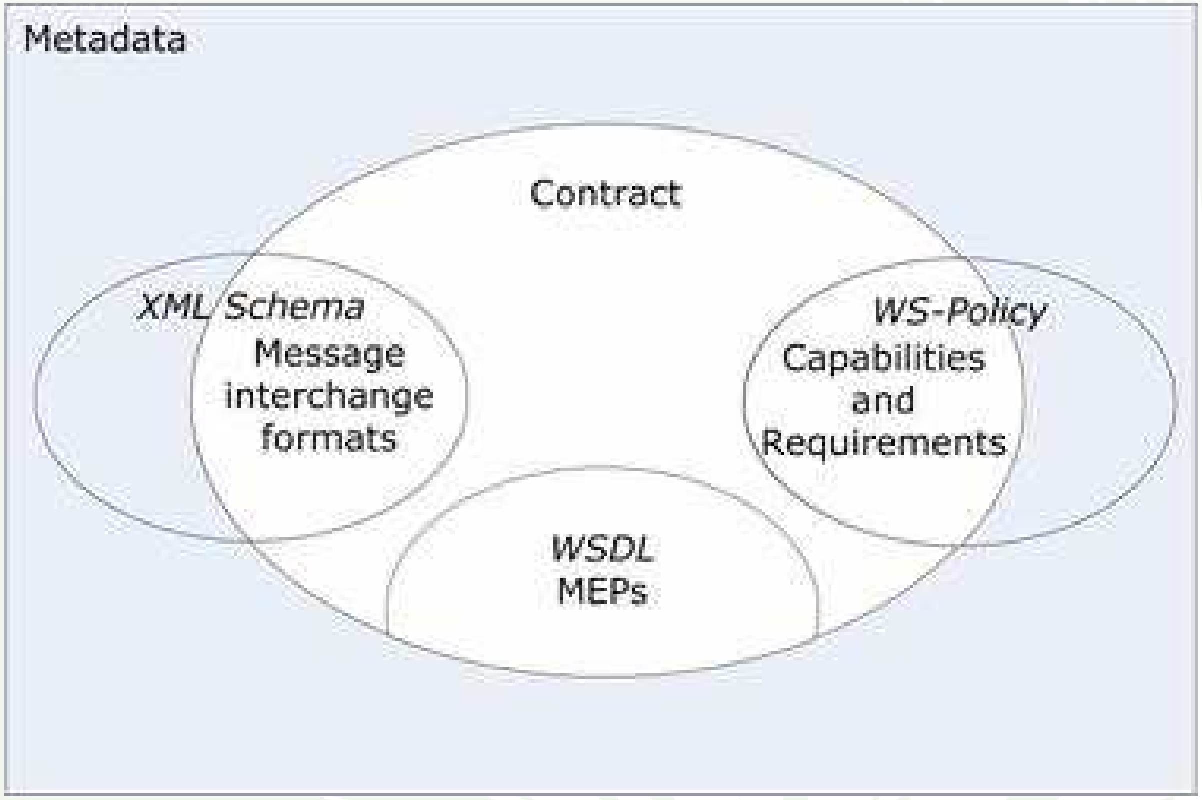
WSDL: Service Contract Starting Point

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions  name ="DayOfWeek"

targetNamespace="http://www.roguewave.com/soapworx/examples/DayOf
Week.wsdl"

xmlns:tns="http://www.roguewave.com/soapworx/examples/DayOfWeek.w
sdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="DayOfWeekInput">
    <part name="date" type="xsd:date"/>
  </message>
  <message name="DayOfWeekResponse">
    <part name="dayOfWeek" type="xsd:string"/>
  </message>

  <portType name="DayOfWeekPortType">
    <operation name="GetDayOfWeek">
      <input message="tns:DayOfWeekInput"/>
      <output message="tns:DayOfWeekResponse"/>
    </operation>
  </portType>

  <binding name="DayOfWeekBinding" type="tns:DayOfWeekPortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetDayOfWeek">
      <soap:operation soapAction="getdayofweek"/>
      <input>
        <soap:body use="encoded"
          namespace="http://www.roguewave.com/soapworx/examples"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
<output>
        <soap:body use="encoded"

namespace="http://www.roguewave.com/soapworx/examples"

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>

  <service name="DayOfWeekService" >
    <documentation>
      Returns the day-of-week name for a given date
    </documentation>
    <port name="DayOfWeekPort"
binding="tns:DayOfWeekBinding">
      <soap:address
location="http://localhost:8090/dayofweek/DayOfWeek"/>
    </port>
  </service>

</definitions>
```

- Sample WSDL file
- Describes port type, binding, input, output, Service name and URL

Source: Rogue Wave

Contract Metadata Beyond WSDL



Source: W3 - http://www.w3.org/2004/08/ws-cc/jsklms-20040903

Sample Service Contract Template

Interface Information

Service Name	1
Service Description	1
Semantic Model	0..1
Number of Data Types	1
Data Types	0..n
Number of Operations	1
Operations	1..n
Service Pedigree	1

Information describing the service, focused on semantics

Security Information

Security Mechanisms	1
Access Criteria & Restrictions	1
Information Security Marking	1

Identifies mechanisms, criteria and restrictions to access the service

Service Level Information

Number of Service Levels	1
Service Level Specifications	0..n
Network Requirements	0..1

Defines level(s) of service provided, and nominal network parameters under which levels of service hold

Implementation Information

Consumer Patterns	0..1
Number of Schedule Dates	1
Schedule	1..n
Number of Operational References	1
Operational Reference	0..n
Versioning Approach	0..1

Provides example design patterns for consumers of the service, schedule dates for the service implementation, and URI to operational info for the service

POC Information

Number Of Contacts	1
Contact	1..n

Information on Points of Contact for this service – such as owner, operational manager, help desk, chief engineer, etc.

ABSTRACT

CONCRETE

Service Access Point Info

Number of SAPs	1
SAP	0..n

Information required to actually bind to the service implementation (or implementations)

Source: MITRE

Meeting Client Needs: MITRE

MITRE

US Federally-Funded Research & Development Center (FFRDC)

Challenge

- Supports government activities around technology, including enterprise architecture
- Needed to provide baseline architectural guidance for Service development
- Insufficient existing examples of Service design artifacts and best practices
- Need to support multi-billion dollar agencies with different change management issues

Solution

- Created the Service Definition Framework (SDF)
- First proposed in 2003 and updated frequently since, latest version in 2007
- Realized that WSDL was insufficient for Service definition
- Included use of OWL-S and semantic information in Service definitions

Results/Benefits

- SDF increasingly widely adopted within different government agencies
- SDF identified as key element of strategy, foundational piece of governance
- Consistent method for describing services across DoD
- Consistent method for finding services across DoD
- Lowered Cost Of Federating Service Discovery Registries

ZapThink is an industry advisory & analysis firm focused exclusively on SOA, EA, and Enterprise 2.0.

Register for an upcoming *Licensed ZapThink Architect* course and obtain your LZA Credential!

Thank You!

Jason Bloomberg

jbloomberg@zapthink.com

Ronald Schmelzer

rschmelzer@zapthink.com