# SOA, Services, APIs, & MQ

or

### How to misuse WMQ without really trying!

# **Table of Contents**

### Background

- Architecture
- Engineering
- Design Principles
- Programming
- Infrastructure

### Service Oriented Architecture (SOA)

- Overview
- IBM's version of SOA
- SOA implications for WMQ
- Services

## Application Programming Interfaces (APIs)

### Summary

## Architecture

## Oxford Dictionary ("modern" OED) definitions:

- "The style in which a building is designed or constructed, especially with regard to a specific period, place, or culture."
- "The complex or carefully designed structure of something."
- "The conceptual structure and logical organization of a computer or computer-based system."

### About the <u>Approach</u>, <u>Design</u> and <u>Structure</u> of a solution.

- Identification of constraints to solution.
- Maximization of <u>Benefits</u> and minimization of <u>Costs</u>.
- Can be formally described and evaluated.
- Widely misunderstood within the IT community.
  - Calling a "position" an architect or a document an "architecture" doesn't make it so!

# **Engineering - 1**

## Oxford Dictionary ("modern" OED) definitions:

- "The branch of science and technology concerned with the design, building, and use of engines, machines, and structures."
- "The work done by, or the occupation of an engineer."
- "The action of working artfully to bring something about".
- Involves both "Design" and "Execution".

### Key Solution Evaluation Concepts

- Cost to Build
- Cost to Maintain
- Time to Market

# **Engineering - 2**

### Key Implementation Concepts

- Final Product Quality
- Total Product Cost
- Time to Market
- Improving any one of these three usually puts pressure on the other two!

### Reliability

- Mean Time Between Failure (MBTF)
- Implications of Failure

### Occam's Razor

- "Entities should not be multiplied unnecessarily".
  - ("Pluralitas non est ponenda sine neccesitate").
- KISS (Keep It Simple Stupid)

# **Design Principles**

Core principles first identified in the 1960s and remain unchanged today!

- Coupling
  - The manner and degree of independence between software modules.
  - Design goal is to have loose coupling.

### Cohesion

- A measure of the strength of relationship between pieces of functionality within a given module.
- Design goal is to have high cohesion.
- Coupling and Cohesion are interrelated and loose coupling tends to induce higher cohesion and vice versa.

### Repackaging of the Original Principles

- SOA (Service Oriented Architecture)
- ▶ S.O.L.I.D.
  - Single-Responsibility "Principle", Open-Closed "Principle", Liskov Substitution "Principle", Interface Segregation "Principle, Dependency Inversion "Principle"
- Etc.

# Programming

### Assemblers

- Assembler instructions mirror computer CPU instructions, there is a one-to-one correspondence between the two.
- In essence, human readable executable binary code.
- Translated into CPU binary code by an "Assembler" program.

### High-Level Languages

- ► Fortran, COBOL, PL/I, C, C++, Java
- Translated into CPU binary code by a "Compiler" program.

### Program Calls / Method Invocations

- Program/Module/Class "A" wants to execute code in Program/Module/Class "B"
  - Local (same server)
  - Remote (server reached through telecommunications stack (e.g. TCP.IP)

#### Data passed between programs

- Call by content (Data itself passed/copied)
- Call by reference (Address of data passed)

# Infrastructure

### Hardware

Computers, Shared storage (disk), Network devices

### Software

- Operating Systems
  - UNIX, Windows, i5/OS, z/OS
- Databases
  - DB/2, Oracle, etc.
- Application Hosting Environments (AHE)
  - Batch/Shell, CICS, IMS, JVMs, Application Servers
- Middleware
  - MQ, Broker, Adapters

### Software Stack

- Hardware
- Firmware (implement virtual instruction set)
- Operating System
- Middleware
- Application

# **Service Oriented Architecture (SOA)**

### History

- First described by Alexander Pasik in 1994 (former Gartner analyst)
- International conferences since 2003
- Promoted by multiple hardware vendors
  - IBM (Rob High Chief SOA Architect)
  - Oracle
  - SAP

### SOA Underpinnings

- Architecture based upon loose coupling and high cohesion
- Architecture lends itself to, but does not require, a distributed solution
- Architecture technology neutral

### Vendor Impact

- Vendors each "bent" the architecture to conform to their product sets
- Vendors co-opted architecture to be a sales vehicle

# **IBM Service Oriented Architecture (SOA)**



# **IBM SOA Layers - 1**

### Interaction Service

- User or Software Interface Layer A true layer!
- WebSphere Portal Server

### Partner Service

- A second User or Software Interface Layer A marketing layer!
- WebSphere Commerce Server

#### Process Service

- Process Layer A true layer! Process logic extracted from business logic.
- Process modeling enabled as an abstraction (BPEL).
- WebSphere Process Server / Business Process Manager.

### Information Service (laaS)

- Abstracting information storage/retrieval A true layer. Abstracting SQL, etc.
- InfoSphere Information Server.

# **IBM SOA Layers - 2**

### Enterprise Service Bus

- Communication, Routing. Transformation A true layer!
- Connections, Asynchronous Messaging, Routing, Transformation
- WebSphere MQ
- WebSphere Message Broker / IBM Integration Bus

### Access Services

- A second piece of the ESB A marketing layer!
- WBIA, Crossworlds

### Business Application Services

- Application Hosting Environments A true layer!
- CICS, WebSphere Application Server























# **SOA using Spaghetti**





# **SOA Objectives**

### Goal #1 - Agility

- Deploy software to market more quickly
  - Reuse infrastructure patterns
  - Minimize impact upon existing applications
  - Modest reuse of new/existing business software
- Concept is to assemble business solutions from available components (e.g. services)
  - Easy access to existing business logic
  - Rapid development of new business logic

### Goal #2 - Cost Reduction

- Minimize new solution impacts upon existing systems
  - Achieved through loose coupling
  - Achieved through high cohesion (reduces/isolates new development)
  - Faster time to market

# **Side Effects of Services**

### Ownership

If software is shared, who owns it?

### Management

- If software is shared, who performs change control?
- Notification?
- Testing?
- Performance?
- Tools

### Monitoring

- How do you predict the shared load on software?
- When will it break?

# **ESB Quality of Service**

### Different ESB implementations have different qualities of service

- WMQ/WMB, WESB, Web Services, ETL
- The implementation chosen should be based on the quality of service required

### Connectivity

- Network connectivity
- Operating System / Runtime environment support
- Programming language support '/ Ease of use

### Integrity

Data Integrity / Transactional support

### Reliability

- Availability (Uptime)
- Monitoring (Proactive Management)

### Agility

- Decoupling of endpoints
- Ease of modification / insertion of new function / transparency to end-points
- Routing and transformation

### Security

# **ESB Implementations**

### Enterprise Service Bus

- Web Services
- Messaging
- Extract, Transform, Load (ETL)

### Web Services

- Fundamentally a Remote Procedure Call
- Tightly coupled point-to-point solution
- Lowest quality of service
- Maximum connectivity

### Messaging

- Loosely or tightly coupled solution
- Highest quality of service: Routing, Transformation, Value-Add processing enabled
- Transactional, requires messaging engine and broker

### ETL

- Alternative to messaging
- Optimized for batch, requires ETL engine

## **Services**

### Every program ever written has an interface.

- Data is passed by content or by reference.
- Interface may be called a "Service".
- Interface may be called an "API".

### Not every program written will be a "Service" with a wide potential.

- Most programs/services are local/tactical rather than strategic/enterprise.
- These programs/services will still be executed in SOA middleware.
- These programs/services will require minimal management.

### Some programs/Services will be Enterprise in nature/potential.

- These programs/services must be managed and controlled.
- In most shops, this is a handful of Services.

### Service Invocation

- Invocation should not be limited by channel (e.g. Web Service vs messaging).
- Invocation should not be limited by runtime environment.
- Invocation should be seen as a "front-end" channel to a service.

### Service Versioning

# **SOA implications for WMQ**

## Decoupling

- "Readers" and "Writers" should know as little about each other as possible.
  - Physical location and object name should be unknown
  - "Readers" and "Writers" should not directly bind to a common object (e.g. queue)
- Many queues used for Point-to-Point messaging in a tightly coupled manner
  - "Reader" and "Writer" both bound to same object (queue)
  - "Reader" bound to object (queue) location (limitation in WMQ)
  - Unable to transparently intercept and manipulate a message
- SOA style usage of WMQ
  - "Readers" and "Writers" each have their own (different) alias
  - Topics & Subscriptions
  - Application created solution for location of messages to be read
- SOA style usage of WMB
  - Applications should "publish" one update
  - Routing and transformation necessary for readers performed in Broker

# **Application Programming Interfaces (APIs)**

## SOA vis a vis API

- SOA Services and API interfaces are identical in function:
  - Allow software to intercommunicate with other known interfaces.
  - Increase capability and speed deployment through component assembly rather than build from scratch.
- SOA Services and API approaches are complementary rather than competitive
- Everything we have been through with SOA is being <u>rediscovered / reinvented</u> in APIs

## SOA

- Generally internal to corporations
- Primarily, but not exclusively, Intranet based
- Generally services are proprietary to corporation

## APIs

- Akin to Open Source
- Primarily Internet based & heavily Web Service oriented

# **APIs and WMQ**

## Existing WMQ Programming Interfaces

- ▶ IBM MQI, AMI, WMQ Classes for Java, WMQ Classes for JMS, .Net, C++
- Sun/Oracle JMS

## Advanced Message Queuing Protocol (AMQP)

- Yet another messaging Interface (aka API)
- Open Standard (OASIS) https://www.oasis-open.org
- https://www.amqp.org
- Interoperability with WMQ will depend upon implementations

## MQLight

- Yet another messaging Interface (aka API)
- Leverages AMQP 1.0 Wire Protocol
- Does not yet interface with WMQ
  - Likely to interface with WMQ through API channels (like SVRCONNs)
  - Stay tuned

# Putting it all together – WMQ, SOA & APIs

### Twenty plus years of WMQ success because of:

- Agility
  - Connectivity across disparate run-time environments.
  - Easy to learn API.
  - Significant speedup of cross platform application deployments
- Data Integrity
- No major transactional competitor when introduced over 20 years ago
  - ETL (Mercator).

### What is now competing with WMQ?

- Web Services; and they are widely used for all of the wrong reasons.
- APIs, and they are the hot new thing (see Web Services).

### APIs

Web Services all over again

# Summary

### WMQ / IBM MQ (iMQ ???)

- Use aliases to decouple "Gets" from "Puts" ("Reader" from "Writers").
- Use Topics and Subscriptions to provide routing and many-to-many connections
  - Remember that a Topic should also be accessed through an alias.

### WMB / IIB

- Applications should only "publish" an event once
- "Writers" (Publishers) should be decoupled from "Readers" (Subscribers)
  - Writer should "publish" result in the standard format
  - Writer should provide all necessary data relevant to the event
- Routing and transformation implemented in Broker

### Enterprise Services

- Should be implemented using WMQ/WMB
- If required, can be reached by a Web Service facade

### APIs

Public facing facades

# **Questions & Answers**



## Presenter

- Glen Brumbaugh
  - <u>Glen.Brumbaugh@TxMQ.com</u>
- Computer Science Background
  - Lecturer in Computer Science, University of California, Berkeley
  - Adjunct Professor in Information Systems, Golden Gate University, San Francisco
- WebSphere MQ Background (20 years plus)
  - IBM Business Enterprise Solutions Team (BEST)
    - Initial support for MQSeries v1.0
    - Trained and mentored by Hursley MQSeries staff
  - IBM U.S. Messaging Solutions Lead, GTS
  - Platforms Supported
    - MVS aka z/OS
    - UNIX (AIX, Linux, Sun OS, Sun Solaris, HP-UX)
    - Windows
    - o iSeries (i5OS)
  - Programming Languages
    - C, COBOL, Java (JNI, WMQ for Java, WMQ for JMS)

