Combining Web Services and The Grid: Enabling a New Way to Think about IT

Kolloquium FU Berlin
(Berlin, Germany, January 14, 2005)
Agenda

The Advent Of Web Services
A Bus For All of IT
Getting Autonomic
Application Structures
On Demand & Implications
Summary

EAI - Example

SAP
Order Entry

ERP

Oracle
Update Available Credit

DB2
Log
High-Value Order

Siebel
Update Customer Record

CRM

CICS
StockHold_Tran

TPM

ERP

DBS
Application Bridges

Application A

A→B Bridge

Application B

Adapters And Queuing: Advent of QoS

Source Environment

Application A

Source Adapter

Message Queuing

Target Adapter

Application B

Target Environment
Flows For Mediation

GET msg FROM queue ORDERValue > 10k€
XFORM into adapter_1 format
STORE msg into DB
PUT msg INTO Q1 (target adapter_1)
APPEND to ErrorLog
SEND eMail to Admin

Mediation: Advent of Brokers

Message xform, Side effects, ...
("Mediation")
Queuing-Based Adapter Architecture

- Application i
- Application j
- Application k

(Source) Adapter
(Target) Adapter
(Target) Adapter

Message Queuing System

Broker

µFlow Engine

Analyze msg & call corresponding µflow

Application Server-Based Connector Architecture

- Application Server
- Source Application
- µFlow Engine
- JMS

Target Application

Source Application
Wrapper: Some Sort Of Adapters

Flows & Channels In B2B Interactions
And So On?

- Well, this integration technology is confusing:
  - Adapters
  - Connectors
  - Wrappers
  - Channels
  - …????
- Where does it stop?

→ Web Services !!!

Web Service Technology: The Key Thing!

1. Web service technology provides a “virtual component model” for using components in a loosely coupled manner

2. When using a Web service the supporting container hides its “middleware idiosyncrasies” (component model behind the implementation of the Web service, the invocation protocol etc.)

3. Web service technology does not provide a new component model for implementing components
Virtualizing Components

Virtual Component

- Web Service

Concrete Component

- (E)JB
- StP
- Assembly

Virtualizing Components

Web Services Usages

RYO Applications

- CICS
- Net
- J2EE

Standard Applications

- ERP
- SCM
- CRM

Client

Data

Content

Tables

© Frank Leymann
The Heart: WSDL (Web Service Description Language)

Agenda

The Advent Of Web Services
A Bus For All of IT
Getting Autonomic
Application Structures
On Demand & Implications
Summary
The Ordering and Guiding Principles

- All standard efforts should foster...
  1. an environment which is
     a. heterogeneous,
     b. distributed,
     c. interoperable,
     d. loosely coupled,
     e. multi-vendor
  2. secure, reliable, recoverable interactions
  3. a single uniform usage model for “components”
  4. business criteria to discover components needed
## Organizing Into A Platform

<table>
<thead>
<tr>
<th>Discovery, Negotiation, Agreement</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choreography</td>
<td>Composite</td>
</tr>
<tr>
<td>Protocols</td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>Atomic</td>
</tr>
<tr>
<td>Reliable Messaging</td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
</tr>
<tr>
<td>Transactions</td>
<td></td>
</tr>
<tr>
<td>Interface + Bindings</td>
<td>Policy</td>
</tr>
<tr>
<td>XML</td>
<td>Non-XML</td>
</tr>
<tr>
<td>Transports</td>
<td></td>
</tr>
</tbody>
</table>

## Quality of Service

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messaging</td>
</tr>
<tr>
<td>Transport</td>
</tr>
</tbody>
</table>

## The Bus And Standards

<table>
<thead>
<tr>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composite</td>
</tr>
<tr>
<td>BPEL</td>
</tr>
<tr>
<td>WS-C, WS-N*,…</td>
</tr>
<tr>
<td>WS-RF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quality of Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messaging</td>
</tr>
<tr>
<td>Transport</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messaging</td>
</tr>
<tr>
<td>Transport</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Messaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliable</td>
</tr>
<tr>
<td>Messaging</td>
</tr>
<tr>
<td>Transactions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transport</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP, TCP/IP, SMTP, FTP, …</td>
</tr>
<tr>
<td>HTTP, TCP/IP, SMTP, FTP, …</td>
</tr>
<tr>
<td>UDDI, WS-Addressing, ADX,…</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composite</td>
</tr>
<tr>
<td>WS-RM</td>
</tr>
<tr>
<td>WS-Security*</td>
</tr>
<tr>
<td>WS-AT, WS-BA,…</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quality of Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messaging</td>
</tr>
<tr>
<td>Transport</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messaging</td>
</tr>
<tr>
<td>Transport</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Messaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security*</td>
</tr>
<tr>
<td>Policy*</td>
</tr>
<tr>
<td>Atomic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transport</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP, TCP/IP, SMTP, FTP, …</td>
</tr>
<tr>
<td>HTTP, TCP/IP, SMTP, FTP, …</td>
</tr>
</tbody>
</table>
Invoking Services Within A Container

Container

Service Bus

Native call

Interop (WS-I)

e.g. SOAP/HTTP

Requestor

"WSDL"

S¹

S²

Quality Of Services & Policies

WS-Policy*

PortType P

Policy

Properties Expected

R

Properties Promised

Policy

WS-Policy*
Matching Endpoints Based On Policies

Selected Port  \(\rightarrow\) Service Discovery \(\rightarrow\) Candidate List

Don't Care! Virtualizing Services

I want ...
But I don't care about a particular provider: Chose the one who is at this point in time "the best" for me!
I need a service of a particular type, but I don’t care about the provider of the service: Anyone will do!

(WSDL, Policies, Data)

Distributed Heterogeneous Environment

Distributed physical clusters and storage
The Grid: Virtualizing Resources

Virtual clusters and storage

→ Service Bus!

OGSA & WS-ResourceFramework

Agenda

The Advent Of Web Services
A Bus For All of IT
Getting Autonomic
Application Structures
On Demand & Implications
Summary
Manageable Resource

Manageability Interface

Manageable Resource

Resource Interface

Allows to plug resource into a management system. It makes the resource manageable (e.g. lifecycle, monitor, metadata, state...)

Resource specific functions, e.g. mountDisk, openDatabase, allocateCPU, ...

Manageability: Sensors And Effectors

Sensors enable monitoring by providing queries ("pull") on properties (actual state, metadata and data) as well as change notification ("push")

Effectors enable explicit state management by providing lifecycle operations and update operations on state and data
Resources Are Web Services 😊

Concept

Manageability Interface

Manageable Resource

Resource Interface

Specification

Port Type

Properties (status, data, metadata)

WS-ResourceProperties

Concept Specification

WS-ResourceProperties

Notification: Base For Monitoring

“Push” sensor!

Notification Source

Subscriber

Notification Sink Y

Subscribe sink Y to events X

Publish notification about X

Deliver notification message

Broker

Bus
Monitor

Receive, aggregate, filter, manage and report metrical & topological information about a manageable resource

Analyze

Correlate data to model complex situations, learn about environment and predict future situations
Plan

Determine and structure the actions needed to achieve goals & objectives described via policies:
Generate a flow!

Execute

Perform and control the execution of a flow
Resource Manager & Autonomic Manager

Autonomic Manager

Knowledge
(policies, state, topology, logs...)

Resource Manager

Management: Using The Bus

Monitor
Operate
Configure

Service Bus

© Frank Leymann
Agenda

The Advent Of Web Services
A Bus For All of IT
Getting Autonomic

Application Structures

On Demand & Implications

Summary

Process-Based Application: Structure
Recursive Aggregation of Web Services

BPEL4WS

pt *

X
Y
P
Z
Q

pt1

pt2

pt3

The Notion Of “Choreography”
Programming Model

Programming in the Large
 Workflow System

Choreography/Flow/Process

Web Services/Functions

Application Server

Programming in the Small

Application

Runtime: The Need For Deployment

Process

Application

Deployment Descriptors

Activity:
- Port Type
- Operation
- BizProperties
- QoS

Ports

Bus
Programming Model: Refinement

Application

Flow

Workflow System

Deployment

DeploymentDesc

Service Bus

Programming in the Small

Components

Application Server

Agenda

The Advent Of Web Services
A Bus For All of IT
Getting Autonomic
Application Structures
On Demand & Implications
Summary
There Is More Needed: Hosting Environment

Computing On Demand

In 3 hours I need the following environment for 45 minutes: 253 machines, 1.7 TB storage, WebSphere, DB2, and the follow ERP applications…
The Ingredients Of On Demand

- Business Processes
- Grid
- Autonomic
- Web Services
- On Demand

Outsourcing
(Information Technology)
Terminology

- **Outsourcing**
  Allocation of IT orders to external companies (e.g. running of [parts of] IT infrastructure including application functionality, development of application functions)

- **Off-Shoring**
  Special case of outsourcing: external company is in foreign country

Situation

- ASP(*) model was never really successful:
  - Model could not exploit economy of scale
  - Interoperability problems
  Technology was not ready for this!

- Recent advancements in Web service-, Grid-, and autonomic-technologies are about to remove these obstructions
  - Recently, outsourcing of both, application functions and business processes is rising sharply
  - Number of intermediate aggregators is rising

➤ Shift from ASP model to Utility model!

(*) Application Service Provider
The Change

- Providers become more cost-effective based on being able to exploit economies of scale
  - Just like traditional public utilities do since decades
  - The key enabler is “dynamic provisioning”
    - I.e. making resources available only when needed
      - “Resources” = hardware, storage, networking, middleware, application functions, business processes

- Consumers get more trust
  - They only have to pay what they use
  - Dependency on third-party – i.e. provider lock-in – is significantly reduced
    - Web service interoperability allows to change providers without major re-write of software
      - Message architecture, security, interfaces, policies, processes, SLAs,…

Reasons For Outsourcing

- They are manifold
- They and their priority change(d) over time
- Cost-savings is mostly ranked high
  - Maintenance of infrastructure and software
    - Purchase cost often not so critical
    - Keeping it current is expensive!
  - Skilled IT personnel is rare
    - Operators, systems management personnel
    - Complexity of IT is constantly raising → Training (costly) to avoid (costly) outages,…
    - Programmers

- Focus on core competencies is important
  - IT is not the proper business of most companies
- Quality improvement is important
  - Application backlog is obstruction to proper functionality
  - Skills of own personnel in some areas not appropriate
Offering Bundles

The “IT artifact bundles” that get outsourced

- **Infrastructure**: Network and/or computing and/or storage
  - Company’s IT staff still run middleware, applications, … on outsourced infrastructure
- **Middleware**: Web pages, security mgmt, …
  - Plus prerequisite infrastructure
- **Application functions**: Individual functions and/or business processes
  - Plus prerequisite middleware and infrastructure

Geographic Distribution of Application

Your and your application
But There Is Hope: Shift of Skills

And There Are Counter-Arguments !!!

- “Off-shoring of infrastructure is high risk”
  - Low-income countries sometimes in political, geographical,… risky areas of the world
  - I.e. higher probability of outages of infrastructure
- “Outsourcing of programming is brain-drain”
  - Company is loosing important skills
  - You become dependent on third party
- “Salaries in low-income countries are raising”
  - Outsourcing/off-shoring is only temporary; try to avoid it
- “Outsourcing/off-shoring of programming has hidden costs”
  - Functional requirements must be documented down to very low level → Lot’s of meetings, reviews,…
    - Increased costs for remaining staff
    - Quality is not up to standards used too
- …
Pragmatics

Reality will be somewhere in between 😊

Agenda

The Advent Of Web Services
A Bus For All of IT
Getting Autonomic
Application Structures
On Demand & Implications

Summary
Conclusion

- Web services is all about “convenient” usage of components in heterogeneous environments
- Programming-in-the-large key for service-orientations
- Autonomic and On Demand based on the above
- Middleware ("The Bus") and tools will make that happen
- Process Models as base of new tradable artifacts
- Socio-political impact
  - Programming-in-the-small can be done in “low-income” countries
  - Outsourcing, off-shoring

The End!

Thanks
For Your Attention...