Application Servers
G22.3033-011

Session 10 - Main Theme
Service Oriented Architectures
Web Services Platforms (Part I)

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

- Continued from Session 8-9:
  - Middleware and Component Infrastructures
  - EAI and B2Bi Environments
- MDA/BPM Technology
  - UML-Based Component Modeling and Related Standards
  - XML Pattern Languages and XML Model-Based Applications
- Web Services
  - XML-RPC
  - SOAP, DIME, and ROPE
  - UDDI, DISCO
  - WSDL
- Summary
- Readings
- Assignment #4
Summary of Previous Session

- .Net Environment (continued)
- Channel Independence and Pervasive Devices
  - Connected Devices
- XML in Component-Based Development Environments
  - XML and JavaBeans/EJBs
  - Deployment Descriptors for OMA Services
  - More on the OMA Persistence Service
- EAI Environments
- B2Bi Environments (e.g., WebMethods Framework)
- BPM Environments
- Summary
- Readings
- Final Project

Application Servers Architectures

- Application Servers for Enhanced HTML (traditional)
  - a.k.a., Page-Based Application Servers
  - Mostly Used to Support Standalone Web Applications
- New Generation Page-Based Script-Oriented App. Servers
  - First Generation Extensions (e.g., Microsoft IIS with COM+/ASP)
  - Servlet/JSP Environments
  - XSP Environment
  - Can now be used as front-end to enterprise applications
  - Hybrid development environments
- Distributed Object Computing Platforms
  - Provide an infrastructure for distributed communications enabling
  - Still need to merge traditional web-oriented computing with object computing
- Object Management Architectures
  - DOC Platform + APIs to reusable services and facilities
- OMA + Component Models -> J2EE, CCM, DNA
- MDAs with XML/Web Services/Channels/BPM-Enabling services
Part I

Middleware and Component Infrastructures
(summary)

Contents

• Motivation
• Middleware
  – Synchronous
  – Asynchronous
• Component Models
  – Application Servers
• Enterprise architectures
  – Business Models and Middleware
## Technology Motivations

### Old Computing
- Prescriptive
- Hierarchical Design
- Determinism
- End result

### New Computing
- Descriptive
- Heterarchical phenomena
- Non-determinism
- Continuing interaction

*Robin Milner, 1991 Turing Award Speech*

## Technology vs. Architecture

- The Long View
- Reference Model of Open Distributed Processing (RM-ODP)
- Map architecture to available technology
- Evolution and addition
  - Tools, materials, …
- Cost of extraction
The Middleware Picture

Three-Tier Client/Server
Middleware Characteristics

- Transparency
- Interoperability
- Portability
- Standard services
- Legacy environment support
- Communication paradigms

<table>
<thead>
<tr>
<th>Middleware</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQSeries</td>
<td>1984</td>
</tr>
<tr>
<td>SunRPC</td>
<td>1988</td>
</tr>
<tr>
<td>TIBCO</td>
<td>1992</td>
</tr>
<tr>
<td>DCE</td>
<td>1992</td>
</tr>
<tr>
<td>COM</td>
<td>1992</td>
</tr>
<tr>
<td>CORBA</td>
<td>1992</td>
</tr>
<tr>
<td>Java</td>
<td>1996</td>
</tr>
<tr>
<td>COM</td>
<td>1996</td>
</tr>
<tr>
<td>XML RPC</td>
<td>1996</td>
</tr>
<tr>
<td>Jini EJB</td>
<td>2000</td>
</tr>
<tr>
<td>.NET SOAP</td>
<td>2000</td>
</tr>
</tbody>
</table>

Transparency

- Access
- Location
- Relocation
- Migration
- Persistence
- Failure
- Replication
- Transaction

See Reference Model of Open Distributed Processing
Interoperability

- Standard protocols
- Semantics
- Data format
- Standards
- Transparency

Portability

- Standard programming interfaces
  - Interface Definition Languages (IDL)
- Increase application market
- Mask network and protocol complexity
- Able to be ported from one platform to another platform with modest and predictable effort.
Services

- Standardized servers
- Defined by an API and supported protocols
- Separation of implementation and definition
  - multiple implementations
- Client and server parts
Legacy Applications

- Logically linked software components
- Mask differences between operating systems, platforms, and network protocols
- Built before middleware
- Plug-and-play
- Interoperability
- Portability
- Coexistence
- Self-managing Entities
- Encapsulation

Communication Paradigms

- Synchronous
  - Remote Procedure Calls
  - Distributed Objects
- Asynchronous
  - Message Oriented Middleware
  - Publish/Subscribe, Event Technologies
Remote Procedure Call

- Addressing
  - name service
- Parameter passing
- Failure
- Security
- Data representation
- DCE, Java RMI, CORBA

Message Oriented Middleware

- Deferred communication
- Complex message semantics
- MQSeries, MSMQ, Java Messaging Service
Event Notification

- Publish and subscribe
- Scalability
- Data backbone
- Asynchronous interaction
- Unsolicited communication
- Scalability
- Heterogeneity
- Addressing - channel, subject, content
- TIBCO, CORBA Notification Service, Elvin

Distributed Objects

- Plug-and-play
- Interoperability
- Portability
- Co-existence
- Self-managing Entities
- Legacy Integration
- Separation of specification and implementation
- CORBA
- Java
- Microsoft .NET
  - DCOM
Components

- Independent deployment
- Third-party composition
- No persistent state
- Contractually specified interfaces
- Late integration

Clemens Szyperski, 1998

---

Component Meta Models

<table>
<thead>
<tr>
<th>Client-Oriented</th>
<th>Microsoft Transaction Server (MTS)</th>
<th>Sun JavaBeans</th>
<th>OMG CORBA Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server-oriented</td>
<td>COM: OLE, OCX, ActiveX</td>
<td>Enterprise Java Beans/J2EE</td>
<td>CORBA Component Model</td>
</tr>
</tbody>
</table>
Java Server Components

Servlets

Enterprise JavaBeans Components

Server Platform:
- JTS
- JMAPI
- JNDI
- JIDL
- JMS
- JDBC

HTTP

IIOP, other protocols

CORBA Component Model

component facet

facets

attributes

receptacles

event sink

event source
Component Architectures

Component A

Component B

Depend upon

Interface C

Are implementations of

Component C1

Component C2

Component C3

Component C4

Component Execution Environment

Application Server

• Core application services
  – Transactions
  – Persistence
  – Security
• Business component container

• Microsoft Transaction Server
  – MTS
• BEA WebLogic, IBM WebSphere
  – EJB, J2EE
• Iona iPortal
Web Services

- Components that interact with one another *dynamically* and use standard Internet technologies
- Advertise the presence of business processes, information, or tasks
  - Universal Description Discovery and Integration (UDDI)
- Simple Object Access Protocol (SOAP)
Part II
(continued from Session 8)

EAI and B2Bi
Enterprise Application Integration
and Business to Business Integration

Also See Session 8 Presentations:
“Enterprise Application Integration (EAI)’’
“Practical Application of EAI’’
“The STP and T+1 Application’’

and the Session 8 Handout on:
“Enterprise Application Integration (EAI)’’

What is Enterprise Application Integration

"EAI is the ongoing process of putting an
infrastructure in place, so that a logical
environment is created that allows business
people to easily deploy new or changing
business processes that rely on IT."

Marc Buyens, Xpragma
Characteristics

- Ongoing process not a product
- Enterprise not IT driven
- Infrastructure at multiple layers
- Logical environment
  - Technical infrastructure not visible
- Abstraction, automation, and flexibility
- Business people “build” business processes
- EAI is difficult, complex, and expensive
- Embrace change
  - not for concrete environments
- Balance gain with pain
- Translate business objectives into actions
  - Assembly of business services

Integration Model

<table>
<thead>
<tr>
<th>Business people</th>
<th>Business Process</th>
<th>Business rules Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT people</td>
<td>Components/Services</td>
<td>EJB</td>
</tr>
<tr>
<td>Infrastructure people</td>
<td>Messaging</td>
<td>MQSeries TIBCO</td>
</tr>
<tr>
<td></td>
<td>Transport</td>
<td>.NET, CORBA, Java/RMI</td>
</tr>
</tbody>
</table>
Business Process Layer

- Translate business flow to processes, tasks, or services
  - Enterprise modeling
- Rules for chaining/interaction of components
- Process modeling, workflow, simulation

OMG Model Driven Architectures

- Mappings from models to middleware
- Choice of implementation technology
- Round trip engineering
- Programmatic access to models
- Enterprise Distributed Object Computing (EDOC)
  - UML, OMG
Components Layer

- Logical objects
- Manipulate logical data (metadata)
- One or more physical objects
- Data transformation, bridging, connectors

Messaging Layer

- Reformatting of messages
- Routing
- Load balancing
- Alternate path switching
- Message warehouse

- Message Broker
Transport Layer

- “Basic” middleware products
  - Message oriented middleware
  - Notification services
- Object Request Brokers (CORBA)
- Transaction monitors
- Audit, logging, and security
- API's, message formats and templates

EAI Approaches – Case Studies

Business Process

Middleware

IBM MQSeries
Crossworlds

- Collaborations
  - Automated business processes, workflows
- Connectors
  - Adaptors to packaged, legacy, or custom components
- Interchange server
  - Business process automation
- Tools
  - System management, process definition, data mapping, transformation and cross-referencing.

Neon Software

• NEON Adaptors
  – Pre-packaged access to off-the-shelf applications and standards

• NEON Accelerators
  – Combination of workflows and data translation for rapid integration (product specific)

• NEON Process Server
  – Business process design tool and execution server

• NEON Integration Servers
  – NEON Impact for creating object/info wrappers
  – NEON Rules for defining data routing
  – NEON Formatter for data translation
  – NEONet for common application integration

NEON Integration

http://www.neonsoft.com
Vitria BusinessWare

- Business Process Management
  - Graphical workflow creation
  - Process modeling and execution
- Enterprise Application Integration
  - Messaging-based connectivity
  - Connectors and data transformation
- B2B Integration
  - Message and document interchange
  - RosettaNet, BizTalk, ebXML
- Real-Time Analysis
  - Monitoring, analysis, and management
- Common runtime services
  - Security, persistence, transactions, etc.

http://www.vitria.com

IBM Websphere/MQSeries

- MQSeries -> WebSphere MQ
- Adaptors
  - including NEON adaptors for MQSeries
- Process Manager (workflow)
- MQ Integrator
  - Rules, Formatter
- MQ Business Integrator
  - B2B + Integrator

MQSeries Integrator

TIBCO

- Active Enterprise
  - Business process integration and automation
- Active Portal
  - Personalized web and wireless access
- Active Exchange
  - B2B integration
  - XML, RosettaNet
- [http://www.tibco.com](http://www.tibco.com)
Active Enterprise

The Future?

- Multi-level programming
- Different abstractions, different languages
- Interoperability not interconnection
- Self-healing, adaptive

- Client-side “containers”
  - Not proxies
Summary

- Middleware
  - Synchronous vs. asynchronous
- Application servers and component models
  - Services and containers
- Enterprise Architectures
  - Enterprise application integration, B2Bi, BPM
  - Multi-level programming

Part III

(Background information)

XML Modeling Capabilities
XML Physical and Logical Structure

- Physical Structure
  - Governs the content in a document in form of storage units
  - Storage units are referred to as entities
  - See http://www.w3.org/TR/REC-xml#sec-physical-struct

- Logical Structure
  - What elements are to be included in a document
  - In what order should elements be included
  - See http://www.w3.org/TR/REC-xml#sec-logical-struct

XML Physical Entities

- Allow to assign a name to some content, and use that name to refer to it
- Eight Possible Combinations:
  - Parsed vs. Unparsed
  - General vs. Parameter
  - Internal vs. External
- Five Actual Categories:
  - Internal parsed general
  - Internal parsed parameter
  - External parsed general
  - External parsed parameter
  - External unparsed general
Logical Structure: Namespaces

- See [Namespaces 1.0](#)
- Sample Element:
  
  ```
  <z:a z:b="x" c="y" xmlns:z="http://www.foo.com"/>
  ```
- Corresponding DTD Declaration
  
  ```
  <!ELEMENT z:a EMPTY>
  <!ATTLIST z:a
    z:b CDATA #IMPLIED
    c CDATA #IMPLIED
    xmlns:z CDATA #FIXED "http://www.foo.com">
  ```

Logical Structure: DTDs

- Shortcomings
  
  - Separate Syntax
    
    ```
    <!ELEMENT Para (#PCDATA) >
    <Para>Some paragraph</Para>
    ```
    vs.
    
    ```
    <ElementType name="Para">
    <ContentModel><PCData/></ContentModel>
    </ElementType>
    ```
  
  - Lack of Support for Data-typing
    
    - DTD Treats an XML Structure as a String of Characters
      
      ```
      <Price currency="USD">1450</Price>
      <Price currency="USD">too high</Price>
      ```
    ```
### Logical Structure: XML Schemas

- **Structures**
  - How elements and attributes are setup in an XML document

- **Datatypes**
  - Built-in datatypes (e.g., String, Boolean, numbers)
  - Generated datatypes (e.g., dates, times, real values)
  - Support for user generated datatypes
  - Backward compatibility with functional subset (DTD)
    - ID, IDREF, NMTOKEN, and SGML-based types

- **Grouping of Elements/Attributes**
  - Archetypes and Attribute Groups

- **Inheritance**
  - Via Basetypes, and Archetypes/Attribute Groups

#### Datatypes Example

```xml
<datatype name='AgeInYears'>
  <basetype name='integer' URI="http://www.w3.org/xmlschemas/datatypes"/>
  <minInclusive>0</minInclusive>
  <maxInclusive>140</maxInclusive>
</datatype>

<attribute name="employeesAge" type="AgeInYears"/>
```
Logical Structure: Navigation

- **URIs/URLs**
  - Syntax for encapsulating a name in any registered namespace, and label it with the namespace
  - Produce a member of the universal set of reachable objects
  - See [http://www.w3.org/Addressing/](http://www.w3.org/Addressing/)
- **XPath**
  - Used to locate certain parts of an XML document
  - See Session 3 handout on “Processing XML documents in Java using XPath and XSLT”

XML Linking/Pointer Language

- **XLink**
  - Allows elements to be inserted into XML documents in order to create and describe links between resources
- **XML Base**
  - Equivalent of HTML BASE functionality generically in XML documents by defining an XML attribute named xml:base
- **Xpointer**
  - Language used as a fragment identifier for any URI-reference that locates a resource of Internet media type `text/xml` or `application/xml`
  - Based on XPath
XLink Example

  Current List of Students
</my:crossReference>`

XPointer Example

- `<!DOCTYPE SPEECH [<!ELEMENT SPEECH (#PCDATA|SPEAKER|DIRECTION)*> <!ATTLIST SPEECH ID ID #IMPLIED> <!ELEMENT SPEAKER (#PCDATA)> <!ELEMENT DIRECTION (#PCDATA)>]>
  <SPEECH ID="a27"><SPEAKER>Polonius</SPEAKER>
  <DIRECTION>crossing downstage</DIRECTION>Fare you well, my lord. <DIRECTION>To Ros.</DIRECTION>You go to seek Lord Hamlet? There he is.</SPEECH>`

- `id(a27).child(2,DIRECTION)
Selects the second "DIRECTION" element (whose content is "To Ros."))`

- `id(a27).child(2,#text)
Selects the second text region,"Fare you well, my lord." (The line break between the SPEAKER and DIRECTION elements is the first text region.)`
XML Base Example

- `<?xml version="1.0"?>`
- `<html xmlns="http://www.w3.org/TR/xhtml1/strict" xml:base="http://somewhere.org">`
- `<head>`
- `<title>Virtual Library</title>`
- `</head>`
- `<body>`
- `<p>`See `<a href="new.xml">what's new</a>!</p>`
- `<p>`Check out the hot picks of the day!</p>`
- `<ol xml:base="/hotpicks">`
- `<li>`<a href="pick1.xml">Hot Pick #1</a></li>`
- `<li>`<a href="pick2.xml">Hot Pick #2</a></li>`
- `<li>`<a href="pick3.xml">Hot Pick #3</a></li>`
- `</ol>`
- `</body>`
- `</html>`

- "what's new" resolves to the URI "http://somewhere.org/new.xml"
- "Hot Pick #1" resolves to the URI "http://somewhere.org/hotpicks/pick1.xml"
- "Hot Pick #2" resolves to the URI "http://somewhere.org/hotpicks/pick2.xml"
- "Hot Pick #3" resolves to the URI "http://somewhere.org/hotpicks/pick3.xml"

XML Data Binding

- Aims to automatically generate substantial portions of the Java platform code that processes XML data
- A Sun project, codenamed “Adelard”
- See JSR-31 XML Data Binding Specification
- see [http://java.sun.com/xml/jaxp-1.0.1/docs/binding/DataBinding.html](http://java.sun.com/xml/jaxp-1.0.1/docs/binding/DataBinding.html)
DTD and XML Schema Design

- Extensibility
  - Reuse existing DTDs or XML Schemas
  - DTDs and XML Schemas must be readable/maintainable

- Compatibility
  - DTDs/XML Schemas updates must be carefully crafted

- Representing Non-Tree-Structured Data
  - Designs must be crafted with efficiency in mind

Canonical XML

- W3C Recommendation (March 15, 2001)
- Canonical XML will be used to represent the result of parsing an XML document
- Canonical XML is necessary to establish the logical “equivalence” of XML documents
- Every well-formed XML document has a unique structurally equivalent canonical XML document
- When “Canonicalizing” an XML document, parsers preserves the minimum information needed by an XML application
## Custom Markup Languages

- Mathematical Markup Language (MathML)
- OpenMath
- Chemical Markup Language (CML)
- Geography Markup Language (GML)
- Wireless Markup Language (WML)
- Synchronized Multimedia Integration Language (SMIL)
- Synchronized Vector Graphics (SVG)
- Extensible 3D (X3D)
- XML-Based User Interface Language (XUL)
- Extensible Log Format (XLF)

## Industry Specific Markup Languages

(see [http://www.oasis-open.org/cover/xml.html#contentsApps](http://www.oasis-open.org/cover/xml.html#contentsApps))

- Ontology Interchange Language (OIL)
- OMG Common Warehouse MetaData Interchange (CWMI)
- OMG Model Driven Architecture (MDA)
- Open Financial Exchange (OFX)
- Straight Through Processing Markup Language (STPML)
- Electronic Commerce Modeling Language (ECML)
- OASIS Business Transactions Technical Committee (OASISBT)
- BizTalk Framework (BizTalk)
- Commerce XML (cXML)
- RosettaNet (RosettaNet)
- Business Process Modeling Language (BPML)
Part IV
(background information)


Also See Session 10 Handout on:
“Secure Message Exchange on the Internet”

XML and Transport Protocols

- Asynchronous Transport Protocols Alternatives
  - Simple Mail Transfer Protocol (SMTP, RFC-821)
    - Simple but no guaranteed delivery and acknowledgment
  - MQSeries
    - Reliable but proprietary and high cost
  - BEEP

- Synchronous Transport Protocols Alternatives
  - Sockets (TCP)
    - Flexible but requires low-level programming
  - HTTP 1.1 (RFC 2068), HTTPR
    - Widely used, supports firewall connections, and security (SSL)
    - Inefficient and session-less (connection- and state-less)
  - CORBA/IIOP
    - Efficient and cross-language/platform compatible but complex
  - SOAP-RP
XML Data Exchange Protocols

- Message Formats Alternatives for Data Exchange
  - Text-based (e.g., EDI, RFC822, SGML, XML)
  - Binary (e.g., ASN.1, CORBA/IIOP)
- Common XML/EDI Interfaces
  - An API that provide a common interface to work with EDI or XML/EDI objects is supported by OpenBusinessObjects
- Guidelines for using XML for EDI
  - [http://www.geocities.com/WallStreet/Floor/5815/guide.htm](http://www.geocities.com/WallStreet/Floor/5815/guide.htm)
  - [http://www.xmledi-group.org/](http://www.xmledi-group.org/)
- Electronic Business XML: ebXML

DOC Application-Level Protocols

- CORBA 3
  - IDL over IIOP v.s. XML over HTTP (XIOP)
    - XML Protocol ([XMLP](http)): XML-Based Messaging Systems
      - Standardized application to application XML messaging (via HTTP, and MQSeries)
- J2EE
  - JMS v.s. XML-based messaging
    - JMS
      - JMS API’s `TextMessage`
      - Custom JMS extensions
        - BEA’s WebLogic `XMLMessage` subclass
    - XML
      - Sun’s Java API for XML Messaging (JAXM)
XML-Based e-Services Protocols

- XML-RPC and Peer-to-Peer Computing
  - [http://xml.coverpages.org/xml-rpc.html](http://xml.coverpages.org/xml-rpc.html)
- Simple Object Application Protocol (SOAP)
  - [http://soap.develop.com/xmlrpc/](http://soap.develop.com/xmlrpc/)
- Universal Description, Discovery, and Integration (UDDI)
- Web Service Definition Language (WSDL)
  - [http://www.w3.org/TR/wSDL](http://www.w3.org/TR/wSDL)

---

XML Fragment Interchange

- Defines a way to send fragments of an XML document without having to send all of the containing document up to the fragment
  - Fragments are not limited to predetermined entities
  - The approach captures the context that the fragment had in the larger document to make it available to the recipient
- See [http://www.w3.org/TR/WD-xml-fragment](http://www.w3.org/TR/WD-xml-fragment)
Organizing Message Data

- Single XML Messages
  - Include everything including binary data into a single XML message
  - Easier to maintain consistency and integrity of constituent parts
  - Popular method is to use Base64 encoding to embed binary data in XML documents
- Separate MetaContent and Resources
  - Send everything at once or let recipient fetch resources as needed

Message Validation and Encodings

- Validation must be performed on messages to accept a fixed set of DTDs or XML Schemas
- Messages must use encodings that are supported by conforming XML processors
  - UTF-8, and UTF-16 provide space efficient alternatives
XML and Security Standards

Security Alternatives
- IPS Security (IPSec, RFC-2401-2412)
  - IP level security, widely used for Virtual Private Networks (VPNs)
- Secure Sockets Layer (SSL)
  - HTTP connection security
  - Transport Layer Security (TLS) standard
- S/MIME (PKCS#7)
  - Secure/Multipurpose Internet Mail Exchange
  - Secure mail standard
  - Uses X.509 certificates for authentication
- Pretty Good Privacy (PGP)
  - De facto standard for e-mail security
  - Can be used to send encrypted digital signatures, and encrypt stored files

SSL Secure Messaging

Confidentiality via symmetric cryptosystem
- Data Encryption Standard (DES, 56-bit key)
- Ron Rivest Code (RC-4, 40-128 bits variable key length)

Integrity via Message Authentication Code (MAC)
- MAC based on secure hash function such as MD5 (Message Digest 5) and SHA 1 (Secure Hash Algorithm 1)

Authentication
- Optional for client (via HTTP/SSL or SSL Certificate-Based), but mandatory for server
- Server’s digital certificate issued by Certificate Authority (CA)
  - Verisign, Inc’s certificates
  - SSL uses X.509 certificate format (ITU-T standard)

Non-Repudiatability (via digital signatures, e.g. PKCS#7)
- DOMHash/Java Cryptography Architecture (JCA)
XML Signatures

- Joint W3C and IETF Working Group
- W3C Candidate Recommendation (April 19, 2001)
- XML compliant syntax used for representing the signature of Web resources and portions of protocol messages
- Procedures for computing and verifying such signatures (I.e., integrity and authentication)
- Does not address encryption, and authorization

Part V

MDA Technology

Also See http://www.omg.org/mda, and associated Webcast and
Session 10 handout on “Application Modeling Using XML”
Model Driven Architectures (MDA)

- MDA Technology Relies on:
  - UML, MOF, and CWM
- Applications Based on MDA are Platform Independent
- Implementations/Realizations can be Targeted to Any Application Server Platform
  - XML-Based Web-Enabled/Web Services-Enabled Platforms
    - Based on CORBA, J2EE, Microsoft

UML’s Business Engineering Methodology

- Business Model/Architecture
  - Use Case View/Model
- Application Model/Architecture
  - Logical and Process View/Models
    - Content, Data, and Process Model (e.g., OIM’s knowledge management, and database/datawarehousing models)
- Application Infrastructure Model/Architecture
  - Implementation View
    - Component Model (e.g., OIM’s component and object model)
- Technology Model/Architecture
  - Deployment View/Model
UML and Modeling Methodologies

- UML: object modeling
- XML: content modeling
- ORM: data modeling

Architecture Development Map

As information is collected, work effort, estimates and solution becomes concrete.
Towards XML Model Based Computing
(review)

- Step 1: Document Object Model
- Step 2: XML Data Binding
- Step 3: Standard XML Information Models
- Step 4: XML Application Services Frameworks
  - Processing, Rendering, Querying, Secure Messaging
- Step 5: XML-Based “Web Object Model”
  - Web Services Architecture
- Step 6: XML Model Driven Architectures (to come)

Current XML-Based Software Development

- Business Engineering Methodology
  - Language + Process + Tools
  - e.g., Rational Unified Process (RUP)
- XML Application Development Infrastructure
  - Metadata Management (e.g., XMI)
  - XML APIs (e.g., JAXP, JAXB)
  - XML Tools (e.g., XML Editors, XML Parsers)
- XML Applications:
  - Application(s) of XML
  - XML-based applications/services
    - MOM & POP
    - Other Services
  - Application Infrastructure Frameworks
XML Metadata Management

- Issue: UML may not provide enough modeling views and enough expressive power in each view to represent a complete application
- Possible Solutions:
  - Extend UML
    - See OIM’s Analysis and Design Model
  - Use Different Modeling Languages:
    - See handout on “XML Information Modeling” (uses different models such as UML, XML, and ORM)
  - Use a Meta-Model: MOF and XMI
    - See handouts on “UML, MOF, and XMI” and “OMG’s XML Metadata Interchange Format (XMI)”

Open Information Model

- Analysis and Design Model
  - Unified Modeling Language (UML) - uml.dtd
  - UML Extensions - umlx.dtd
  - Common Data Types - dtm.dtd
  - Generic Elements - gen.dtd
- Components and Object Model
  - Component Description Model - cde.dtd
- Database and Warehousing Model
  - Database Schema Elements - dbm.dtd
  - Data Transformation Elements - tfm.dtd
  - OLAP Schema Elements - olp.dtd
  - Record Oriented Legacy Databases - rec.dtd
- Knowledge Management Model
  - Semantic Definition Elements - sim.dtd
XML Support for UML Modeling

- Meta Object Facility (MOF)
  - CORBA Common Facility for the Management of Meta Information such as UML Models, Database Schemas, Programming Language Types, etc.

- XML Metadata Interchange (XMI)
  - Enables interchange of metadata between modeling tools
  - Develop models using Rational Rose or Java, and the XMI toolkit
  - Use Objects by Design xmi-to-html.xsl style sheet and Cocoon framework to present the resulting model

MDA Development Approach

![Diagram of MDA Development Approach]
Sample MOF MetaModel

Sample MOF MetaModel
(continued)
MOF to XML Mapping

Simplified UML XMI DTD
Model Driven Development Tool
(http://www.sygel.com/)

XML Information Modeling

- Steps
  - Documenting the Information Structure
  - Representing the Information Structure in XML Form
  - Defining XML DTDs and/or Schemas

- Modeling Techniques
  - UML: object modeling
  - XML: content modeling
  - ORM: data modeling
  - See Session 3 handout on “XML Information Modeling”

- UML, MOF and XMI
  - See handout
Using UML to Model MLs

Glossaries

<<XSDcomplexType>>

entry [0..1] : string

authorName : string

creationDate : date

DialectInfo

frenchCanadian : string

dialectCode : DialectCode

<<XSDcomplexType>>

DialectCode

from XSD Datatypes

<<XSDcomplexType>>

SignedEntry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)

<<XSDcomplexType>>

Entry

english [0..1] : string

french [0..1] : string

Glossary

<<XSDelement>>

topic : string

<<XSDattribute>>

creationDate : date

Glossaries

<<XSDcomplexType>>

0..n +glossary 0..n +entry 0..n

PlaceHolder

(from XHTML)
XML Metadata Management

(continued)

- Practical Use of XMI:
  - Develop a model using Rational Rose or Java, and the XMI toolkit
  - Use Objects by Design xmi-to-html.xsl style sheet and Cocoon framework to present the resulting model

- Current State of Tools:
  - Forward-Engineering of UML into XML Schemas is possible
    - Can export .mdl files from Rational Rose, and convert them into XMI using SoftModeler (www.softera.com)
    - SoftModeler can convert UML into XML W3C Schemas
  - Reverse-Engineering of XML Schemas into UML is more difficult
    - Not supported by current tools

Part VI

Web Services Technology

Also See: http://ejbinfo.com/articles/00/10/28/0933234.shtml
and http://www.clipcode.com/peer/beep_protocol_guide.htm
Towards Web Services ...

- The Web Object Model
  - Adding “methods” to XML (e.g., WIDL)
  - SOAP
- XML and UML based component modeling
  - Model-based applications via UML to EJB component binding
  - XML Metadata Interchange (XMI) and related technology
- XML and Model-based computing
  - Model-based computing frameworks

Towards a “Web Object Model”

- See dated article by Frank Manola (OBJS)
- Should compare Manola’s proposed approach for building a web object model to the approach suggested in today’s Web Services architecture
- Web Services use XML as a portable data language to interchange information
  - The binding to Java and/or other application languages is implemented via SOAP endpoint connectors (e.g., SOAP to JMS)
A Vision…

• A peaceful and harmonious services-based world…
  – Applications everywhere working together through shared, open services
  – Seamless, easy integration between applications, across departments, across organisations, across the world
  – One common set of standards everywhere

• A programmable Web…
The Vision

Web Services Vision
A Programmable Web?

• What’s wrong with the Web today?
  – Really just same old dumb client model
• The services-based Web
  – Servers offer services not just pages
  – Smart clients weave new applications by combining and integrating services
  – As significant change.
  – A Web for computers/enterprises as well as people.

Alternatives?

• Screen-scraping analogues
  – Programming against human interfaces
  – Divining the semantics of Web pages?
  – Fragile, clumsy applications
  – Web Services is simpler solution
    • Services to return available seats, place order, update account, …
    • No semantics, no untangling formatting
The Web Today

• * Content is dynamic
• Today, sites provide up-to-the-minute content, incorporating news headlines, stock quotes, and customized pages
• * Bandwidth is cheaper
• * Storage is cheaper
• * Pervasive computing is becoming more important.
• With hundreds of millions of devices such as mobile phones, pagers, and palmtop computers already in use, we're reaching the end of the days when PCs are the most common device on the Internet. As platforms become more diverse, technologies like XML become even more important.

Where Web Services fit in

• * Content is dynamic.
• A Web service has to be able to combine content from many different sources. That may include stock quotes, weather reports, or news headlines. In a legacy environment, that content may include inventory levels, purchase orders, or directory information, all from back-end systems.
• * Bandwidth is cheaper.
• A Web service can now deliver types of content (streaming video or audio, for example) unthinkable a few years ago. As bandwidth continues to grow, Web services must adapt to new content types.
Web Services, cont’d.

• * Storage is cheaper.
• A Web service must be able to deal with massive amounts of data intelligently. That means using technologies such as databases, LDAP directories, caches, and load balancing software to make sure that scalability isn't an issue.
• * Pervasive computing is becoming more important.
• A Web service can't require that users run a traditional browser on some version of Windows. Web services have to serve all sorts of devices, platforms, and browser types, delivering content over a wide variety of connection types.

Web Services are a new Web Model

• New Model:
  – transactions are initiated automatically by a program, not necessarily via a browser
  – services can be described, published, discovered, and invoked dynamically
  – new ways of using the web: intelligent agents, marketplaces, auctions
• Extending the client-server model:
  – invoke “methods” on remote objects
  – use a directory of services and service providers
  – employ standard description languages based on XML
Examples of Web Services

• Business Information with rich content
  – weather reports, news feeds, credit checks, requests for quote, auctions, ...
• Transactional web services
  – reservations, supply chain management, purchase orders
• Business processes
  – workflow

Web Services

• Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web.
• Web services perform functions, which can be anything from simple requests to complicated business processes.
• Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.
Web Service Components

- Service Provider: publishes and provides
- Service Registry: support for publishing and locating
- Service Requestor: finds services and binds services to service provider

Web Service Operations

- * Publish/Unpublish: advertising services to a registry (publishing) or removing those entries (unpublishing). The service provider contacts the service broker to publish or unpublish a service.
- * Find: performed by service requestors and service brokers together. The service requestors describe the kinds of services they're looking for, and the service brokers deliver the results that best match the request.
- * Bind: takes place between the service requestor and the service provider. The two parties negotiate as appropriate so the requestor can access and invoke services of the provider.
Web Services, cont’d.

• The standards for expressing and using Web services are based on XML and HTTP
• There are three key extensions:
  – Service registration
    • WSDL: Web Services Definition Language
  – P2P interaction
    • SOAP: Simple Object Access Protocol
  – Service discovery
    • UDDI: Universal Discovery and Description Interchange

XML Web Services

• XML-based Network Computing
  – XML messages in the protocol
  – XML definitions of services
  – Pass across XML documents
  – Building on other XML standards
  – All ‘human readable’
• Possibly running over HTTP…
  – And possibly not as well
XML is the Key to Business Integration

- Heterogeneous system environments and platforms
- Wide range of applications: transactional and collaborative
- Wide range of business processes, intra- to inter-enterprise
- People need universal access anywhere, anytime
- Data needs to be able to be customized

e-Business Future

- XML is the way to exchange data
  - between platforms
  - regardless of the programming language
  - on different devices
  - with different middleware
- Web services require STANDARDS for exchanging data
  - SOAP: XML protocol for Request and Response
- The SOAP protocol is very simple - Remote Procedure Calls
  - the added power comes from Internet standards for remote object invocation
ebXML

- Electronic Business XML
  - electronic data interchange (EDI)
  - facilitate trade and exchange of data within particular industries
  - technical drawings, specifications, transactions
  - cross industry and within a vertical (i.e. transport) industry
  - use SOAP as message envelope

Web Services and Network Computing?

- Web Services are just XML-based network computing protocols
  - Nothing inherently new
  - Similar to EDI, CORBA, DCOM, RMI, CICS, MQ, …
  - More verbose and slower
  - Perhaps somewhat simpler
    - This may be fixed as standards evolve…
  - Currently compatible
    - One set of common standards
Previous Attempts?

- CORBA standards
  - Major changes needed to existing applications and infrastructure
  - Now a niche technology (killed by J2EE)
- Microsoft’s COM+, Sun’s J2EE
  - Any Windows platform you like… or
  - any language you like, as long as it’s Java
  - Application rewriting or adaptors
- Limited interoperability
- Not designed for the public Internet

Standards?

- Many ‘standards’ groups
  - Largely committees of industry players representing vendor interests
  - OMG, OASIS, OSF, … + W3C
  - Temporary alliances against a common enemy (often Microsoft and/or IBM)
- De facto standards (proprietary)
  - Owned/controlled by one vendor
    - Microsoft products, IBM products, Sun’s J2EE specs
    - Can be more focussed - need for product
Standards

• Formal standards bodies
  – W3C
  – IETF
  – ECMA

• All types of standards processes active in Web Services
  – Especially now that the core proposals from MS and IBM have won through
  – Heavily commercial & competitive processes

Standards…

• Software standards can specify anything
  • And may not be easily implemented
  • And can suffer from committee syndrome

• Software standards can expand forever
  • Already too many WS standards?
  • Already too complex? Too many players?
  • Hard to argue against standardization though – obviously a ‘good thing’
Microsoft .NET Platform
Smart Devices + Windows XP + Web Services

- [http://www.microsoft.com/net](http://www.microsoft.com/net)
- First set of Microsoft Web Services

**XML Web services**

- XML Request
- XML Response

**Web Services**

- XML – meta language
- SOAP – access protocol
- WSDL - description
- UDDI - discovery

Not B2B suitable
WS-Security ongoing
WS-Choreography started
Synchronous
SOAP

- Lightweight protocol for information exchange
  - Envelope
    - Framework that describes a message and its processing
  - Set of encoding rules
    - Used to express instances of application datatypes
  - Convention
    - RPC calls and responses
- Apache distributes an implementation of SOAP referred to as “AXIS”

Content, Discovery, Universal access, and Intelligent Software Agents

- UDDI: Universal Description, Discovery, and Integration
  - Industry-wide effort to bring a common standard for business-to-business (B2B) integration
  - Set of standard interfaces for accessing a database of web services
  - jUDDI (pronounced "Judy") is an open source Java-based implementation of a UDDI registry
  - Also see
    - [http://uddi.microsoft.com/default.aspx](http://uddi.microsoft.com/default.aspx)
    - [http://www.oasis-open.org/cover/uddi.html](http://www.oasis-open.org/cover/uddi.html)
    - [http://www.itpapers.com/cgi/SubcatIT.pl?scid=436](http://www.itpapers.com/cgi/SubcatIT.pl?scid=436)
- Intelligent Software Agents: ATLAS, Aglets, etc.
XML-Based e-Services Protocols and Architectures

- XML-RPC and Peer-to-Peer Computing
  - [http://xml.coverpages.org/xml-rpc.html](http://xml.coverpages.org/xml-rpc.html)
- Simple Object Application Protocol (SOAP)
  - [http://soap.develop.com/xmlrpc/](http://soap.develop.com/xmlrpc/)
- Universal Description, Discovery, and Integration (UDDI)
- Web Service Definition Language (WSDL)
  - [http://www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl)
- Pervasive devices
- Resource Description Framework (RDF)
  - Platform for Internet Content Selection (PICS)
  - Platform for Privacy Preferences (P3P)
  - Channel Definition Format (CDF)
  - Rich Site Summary (RSS)
  - Blocks Extensible Exchange Protocol (BXXP)

XML-Based e-Services Protocols and Architectures (continued)

- XML Protocol (XMLP): XML-Based Messaging Systems
  - Standardized application to application XML messaging (via HTTP, and MQSeries)
- XML and User Identification/Security
- XML and Databases
  - XML and JDBC
  - XML Extensions and Tools for Oracle, Informix, IBM DB2, and Microsoft SQL Server
- Transaction Authority Markup Language (XAML)
  - Coordinated processing of transaction-supporting web services between internal fulfillment services and external services
XML-Based e-Services Protocols and Architectures (continued)

- Sun’s Open Net Environment (ONE)
- HP’s NetAction/e-speak platform
- Oracle’s Dynamic Services platform
- Microsoft .NET platform
- IBM WebSphere Architecture (WSA) platform

Network Computing 101

- Find something
  - Object, component, service
- Find out how to call it
  - What information it needs
  - Where it is located
- Ask it to do something for you
  - Call a method, invoke a service
- And add a few useful services
Web Services Infrastructure, cont’d.

- Information has to be able to flow
  - between different applications
  - using different programming environments
  - on different hardware
  - running different operating systems
  - using different communication protocols
- Loose coupling is desirable
  - greater scalability, greater choice of vendors, better maintainability, and fewer dependencies

Web Services Stack
XML and J2EE
(release chronology)

- Java XML Pack
  - All in one XML technology for Java
  - e.g., SAX, DOM, XSLT, SOAP, UDDI, ebXML, and WSDL
- Java XML Pack Summer 02 Release
  - Java API for XML Messaging (JAXM 1.1)
  - Java API for XML Processing (JAXP 1.2)
  - Java API for XML Registries (JAXR 1.0_01)
  - Java API for XML-based RPC (JAX-RPC 1.0)
  - SOAP with Attachments API for Java (SAAJ 1.1)
- Java Web Services Developer Pack 1.1
Web Services Stack

Conceptual Architecture

Discovery Agencies
- Discovery
- Publication
- Inspection

Description
- Business Level Agreements
- Service Level Agreements
- Composition
- Orchestration
- Presentation
- Policy
- Implementation Description
- Interface Description
- XML Schema

Wire
- Extensions
- Packaging
- Transport

QoS
- Security Mgmt

Implementation Diagram

Misc.
- Workflow Language
- Security

Transport (the wire)
- SOAP Messages & Headers
  - Reliability
  - Security
  - Asynchrony
  - Routing

Description
- SOAP Messages & Headers
  - Caching

Discovery
- WSDL
  - Service Characteristics
  - Choreography
  - XML Schema
  - Registry
  - Inspection
### Derivative Architecture Patterns

#### Web Services Latest Specifications


- **Baseline WS Specifications**
  - SOAP, UDDI, WSDL
- **BPEL4WS**
  - Business Process Execution Language for WS
- **Global WS Specifications**
  - WS-Coordination
  - WS-Inspection
  - WS-Referral
  - WS-Routing
  - WS-Security
  - WS-Policy
  - WS-Transaction
IBM WebSphere SDK for Web Services V. 5.0

- Services-Oriented Computing
  - New approach to B2Bi
  - XML message encoding architecture + HTTP as a communication transport
  - Web services are about how to access a business service, while J2EE is about how to implement that business service

- Web Services with WSAD Demonstration Tutorials:
  - Build and Test
  - Deploy and Publish
Generic Web Services Architecture

XML Request

Listener

Business Facade

XML Response

Business Logic

SOAP/UDDI Model

eXtensible Markup Language (XML) Model

UDDI registry

eXtensible Markup Language (XML) registry and repository

Service discovery

HTTP server

SOAP processing node

SOAP/HTTP server or server-based application

WSDL interface definition

Company A
Web Service Provider

SOAP request

SOAP response

Company B
Web Service Requester

XML request

XML response
Basic Steps with Web Services

1. A service provider creates a Web service.
2. The service provider uses WSDL to describe the service (to a UDDI registry).
3. The service provider registers the service (in a UDDI registry and/or ebXML registry/repository).
4. Another service or user locates and requests the registered service by querying UDDI and/or ebXML registry(ies).
5. The requesting service or user writes an application to bind the registered service using SOAP (in the case of UDDI) and/or ebXML.
6. Data and messages are exchanged as XML over HTTP.

Finding Things

- UDDI - a directory of services
  - Universal Description, Discovery and Integration
  - A Web Service in its own right!
  - White, Yellow and Green ‘pages’
- Vision of a single universal directory?
  - Reality is probably many directories
  - Accepted but hasn’t taken off… yet
  - Included with Windows Server 2003
    - Intended for internal use only (repository)
UDDI Pages

• *White Pages:* These provide general information about businesses, such as names, addresses and contact details.
• *Yellow Pages:* Similar to yellow pages. Let clients search for businesses that provide services to particular industry segments or offer specific services. A number of taxonomies are available for UDDI classification and they include the NAICS-1997 (North American Industry Classification System) and the UNSPSC-7.03 (Universal Standard Products and Services Code).
• *Green Pages:* Technical information on services being offered, including how to call them and where they are located. This information is returned as a ‘type model’ (tModel) which is most likely the WSDL description for the service. The tModel includes information about the location of the service, the service name, the operations it supports and their parameters. Businesses can define their own tModels to describe their services, but they can also take advantage of common sets of services defined by industry standards bodies.

UDDI

• *Universal Description, Discovery, and Integration*
  – UDDI servers act as a directory of available services and service providers
  – SOAP can be used to query the UDDI for services
**UDDI**

- **UDDI provides three basic functions:**
  - Publish: How the provider of a Web service registers itself.
  - Find: How an application finds a particular Web service.
  - Bind: How an application connects to, and interacts with, a Web service after it's been found.

- **A UDDI registry contains three kinds of information, described in terms of telephone directories:**
  - White pages: Information such as the name, address, telephone number, and other contact information of a given business.
  - Yellow pages: Information that categorizes businesses. This is based on existing (non-electronic) standards (see Resources).
  - Green pages: Technical information about the Web services provided by a given business.
UDDI, cont’d.

- Four information types that are important in UDDI:
  - Business information: a `BusinessEntity` object, which in turn contains information about services, categories, contacts, URLs, and other things necessary to interact with a given business.
  - Service information: Describes a group of Web services. These are contained in a `BusinessService` object.
  - Binding information: The technical details necessary to invoke a Web service. This includes URLs, information about method names, argument types, and so on. The UDDI4J `BindingTemplate` object represents this data.
  - Information about specifications for services: This is metadata about the various specifications implemented by a given Web service. These are called `tModels` in the UDDI specification; the UDDI4J `TModel` object represents this data.

Finding Out About Things

- WSDL - Web Services Description Language
- XML description of a service
  - Interfaces, methods, parameters
  - Request, response
  - How to bind to the service
  - Widely accepted standard
- Generated and consumed by tools
  - Microsoft Visual Studio, IBM WebSphere
WSDL

- The overall structure of requests, the relevant data types, the schema of the XML elements used, and other such matter are left to the trading partners by the SOAP specification itself.
- WSDL provides a standard for service specification that unites the types of requests and the requirements needed to process them.
- Four forms of operations have built-in support in WSDL: one-way, request-response, solicit-response, and notification.
- WSDL supports unidirectional (one-way and notification) and bidirectional (request-response and solicit-response) port types. Faults are only supported in the bidirectional port types.

```xml
<?xml version="1.0"?>
<definitions name="StockExchange">
  <types>
    ...
  </types>
  <message name="getStockQuoteRequest">
    <part name="symbol" type="xsd:string"/>
  </message>
  <message name="getStockQuoteResponse">
    <part name="Result" type="xsd:float"/>
  </message>
  <portType name="StockQuoteService">
    <operation name="buyRequest" parameterOrder="symbol">
      <input message="wsdlns:getTradePriceRequest"/>
      <output message="wsdlns:getTradePriceResponse"/>
    </operation>
  </portType>
  <binding>
    ...
  </binding>
</definitions>
```
WSDL Example

• A WSDL description of our service for a Snowboarding endorsement query.

• <definitions> describes a set of related services.

• <types> allows the specification of low-level data-typing for the message or procedure contents.

• <portType> groups messages that form a single logical operation. We can have an EndorsingBoarder request which triggers an EndorsingBoarder response, or an EndorsingBoarderFault. This particular exchange is grouped together into a WSDL port type. The relationship to messages is made by qualified name reference.

WSDL, cont’d.

• <message> defines the data format of each individual transmission in the communication. One message represents the EndorsingBoarder request and the other the response.

• In the following example, this is a “simple” statement that the body of the message is a particular element from the schema in the types section. The breaking of a transmission into message parts depends on the logical view of the data. For instance, if the transmission is a remote procedure call, the message might be divided into multiple parts, one of which is the procedure name and metadata and the rest of which are the procedure parameters. There is naturally a relationship between the granularity of the data-typing and the break-down of the message into parts.
WSDL Example

```xml
<?xml version="1.0"?>
<definitions name="EndorsementSearch"
    targetNamespace="http://namespaces.snowboard-info.com"
    xmlns:es="http://www.snowboard-info.com/EndorsementSearch.wsdl"
    xmlns:esxsd="http://schemas.snowboard-info.com/EndorsementSearch.xsd"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <schema targetNamespace="http://namespaces.snowboard-info.com"
        xmlns="http://www.w3.org/1999/XMLSchema">
      <element name="GetEndorsingBoarder">
        <complexType>
          <sequence>
            <element name="manufacturer" type="string"/>
            <element name="model" type="string"/>
          </sequence>
        </complexType>
      </element>
      <element name="GetEndorsingBoarderResponse">
        <complexType>
          <all>
            <element name="endorsingBoarder" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="GetEndorsingBoarderFault">
        <complexType>
          <all>
            <element name="errorMessage" type="string"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>
  <message name="GetEndorsingBoarderRequest">
    <part name="body" element="esxsd:GetEndorsingBoarder"/>
  </message>
  <message name="GetEndorsingBoarderResponse">
    <part name="body" element="esxsd:GetEndorsingBoarderResponse"/>
  </message>
</definitions>
```
<portType name="GetEndorsingBoarderPortType">
  <operation name="GetEndorsingBoarder">
    <input message="es:GetEndorsingBoarderRequest"/>
    <output message="es:GetEndorsingBoarderResponse"/>
    <fault message="es:GetEndorsingBoarderFault"/>
  </operation>
</portType>

<binding name="EndorsementSearchSoapBinding" type="es:GetEndorsingBoarderPortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetEndorsingBoarder">
    <soap:operation soapAction="http://www.snowboard-info.com/EndorsementSearch"/>
    <input>
      <soap:body use="literal"
        namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/>
    </input>
    <output>
      <soap:body use="literal"
        namespace="http://schemas.snowboard-info.com/EndorsementSearch.xsd"/>
    </output>
  </operation>
</binding>

<service name="EndorsementSearchService">
  <documentation>snowboarding-info.com Endorsement Service</documentation>
  <port name="GetEndorsingBoarderPort" binding="es:EndorsementSearchSoapBinding">
    <soap:address location="http://www.snowboard-info.com/EndorsementSearch"/>
  </port>
</service>
Calling Things

• SOAP – Simple Object/ Services- Oriented Access Protocol
  – Originally simple remote procedure calls using XML-formatted protocol
    • XML-RPC is a subset of SOAP
    • Can pass messages as well as RPC
  – Envelope, header, body
  – Extensible protocol
    • Add new header elements to access additional semantics
  – Very widely accepted stand—

SOAP

• SOAP
  – substitutable transport bindings (not just HTTP)
  – multiple language bindings
  – substitutable data encodings
  – independent of programming language, object model, operating system, or platform
SOAP

• SOAP is a “lightweight” protocol for exchange of information in a decentralized, distributed environment.
• XML based and consists of three parts:
  • an envelope that defines a framework for describing what is in a message and how to process it,
  • a set of encoding rules for expressing instances of application-defined datatypes, and
  • a convention for representing remote procedure calls and responses.
• SOAP can potentially be used in combination with a variety of other protocols; however, presently SOAP is used with HTTP and the HTTP Extension Framework.

SOAP Message Structure

• Request and Response messages
  • Request invokes a method on a remote object
  • Response returns result of running the method
• SOAP defines an envelope
  • envelope wraps the message itself
  • message is in a different vocabulary
  • namespace prefix is used to distinguish the two parts
SOAP

- **Envelope (Mandatory)** - Marks the start and end of a message
- **Header (Optional)** - General information about message – e.g. authentication and transaction management
- **Body (Mandatory)** - Data for the actual message or document being sent

SOAP, cont’d.

- SOAP hides the service implementation from the requester
- “Simple”, standard XML messages for request and response
SOAP Example

• In natural language, the next slide encapsulates the question "Which professional snowboarder endorses the K2 FatBob?"

• In natural language, the response encapsulates the simple string response "Chris Englesmann".

SOAP, cont’d.

• POST /EndorsementSearch HTTP/1.1 Host: www.snowboard-info.com Content-Type: text/xml; charset="utf-8" Content-Length: 261 SOAPAction: "http://www.snowboard-info.com/EndorsementSearch"


• <SOAP-ENV:Body>

• <m:GetEndorsingBoarder xmlns:m="http://namespaces.snowboard-info.com"> 
  <manufacturer>K2</manufacturer> <model>Fatbob</model>
  </m:GetEndorsingBoarder> </SOAP-ENV:Body> </SOAP-ENV:Envelope>
SOAP Response

  - `<SOAP-ENV:Body> <m:GetEndorsingBoarderResponse xmlns:m="http://namespaces.snowboard-info.com">
    <endorsingBoarder>Chris Englesmann</endorsingBoarder>
  </m:GetEndorsingBoarderResponse>
  - </SOAP-ENV:Body>
- </SOAP-ENV:Envelope>

RESPONSE

Another SOAP Example

- In natural language, the next slide encapsulates the question “What is the last trade price for this stock?”

- The response is 34.5.
Everything Else

- SOAP, WSDL (& UDDI) only meet simplest requirements
  - Still usable for many internal or non-critical applications
    - Google searches, MapPoint maps
- Other standards emerging for everything else
  - Security, transactions, routing, reliability, attachments, …
  - Current standards battleground…
Benefits of Web Services

• 1. promote interoperability
  – Interaction between a service provider and a service requester is completely platform and language independent.
  – This interaction requires a WSDL document to define the interface and describe the service, along with a network protocol (usually HTTP).
  – Because the service provider and the service requester have no idea what platforms or languages each other are using, interoperability is a given.

Benefits of Web Services, cont’d.

• 2. enable just-in-time integration
  – As service requesters use service brokers to find service providers, the discovery takes place dynamically. Once the requester and provider have found each other, the provider's WSDL document is used to bind the requester and the service together.
  – This means that requesters, providers, and brokers work together to create systems that are self-configuring, adaptive, and robust.
Benefits of Web Services, cont’d.

• 3. reduce complexity through encapsulation
  – Service requesters and providers concern themselves with the interfaces necessary to interact with each other.
  – A service requester has no idea how a service provider implements its service, and a service provider has no idea how a service requester uses its service. Those details are encapsulated inside the requesters and providers. That encapsulation is crucial for reducing complexity.

Benefits, cont’d.

• 4. give new life to legacy applications
  – Take an existing application, generate a SOAP wrapper, then generate a WSDL document to cast the application as a Web service. This means that legacy applications can be used in interesting new ways.
  – The infrastructure associated with legacy applications (security, directory services, transactions, etc.) can be "wrapped" as a set of services as well.
Security!

• “SOAP is firewall friendly”
  – Normally transported over HTTP
  – Firewalls expect HTTP to be Web requests, not procedure calls
    • Fetch the brochure…
    • … not update my bank account…
  – This is probably a good thing!
    • Have to rely on strong security rather than weaker physical security

WS-Security

• Proposal from IBM & MS
• SOAP header extensions
  – Protect integrity and confidentiality of messages
  – Attach encoded security tokens
    • X509, Kerberos
• Leverage XML standards
  – XML Signature
  – XML Encryption
More WS-Security

• WS-Policy
  – Specifying requirements & capabilities
  – Policies including privacy, encoding security tokens, QoS, …
  – More than just security…
  – First published Dec 2002

• WS-Trust
  – Establishing trust relationships
  – Trusted security tokens
  – First published Dec 2002

More WS-Security

• WS-Privacy
• WS-SecureConversation
• WS-Federation
• WS-Authorization

• And probably a few more…
Yet More Security Standards

• SAML
  – Representing authentication and authorisation information

• XACML
  – Access Control

• XrML
  – Digital Rights (similar to access control)

Yet More Security

• All these standards built on existing strong security standards
  – PKI
  – Kerberos
  – Digital signatures

• And why so much interest?
Security Concerns?

WS Transactions

- Ensuring consistency and integrity
  - WS-Coordination & WS-Transactions
    - Proposals from MS and IBM (& BEA)
    - ACID and long-running transactions
  - BTP
    - Proposal from OASIS (Oracle, Sun, BEA, …)
  - No consensus yet… but IBM/MS will probably win…
Reliability

• Reliable message delivery
  – Guaranteed, exactly-once, in-order
  – Despite failures in applications, systems and network
• WS-ReliableMessaging
  – IBM and Microsoft
• WS-Reliability
  – Sun, Oracle, etc…

Business Processes

• WS standards for how businesses interact and work together
  – Workflow language models
  – BPEL4WS (IBM & Microsoft + partners)
    • Merging of WSFL and XLang
  – Competing standards from Oracle, Sun, HP through OASIS
### Transports

- **Web Services independent of HTTP**
  - HTTP only offers partial solution
    - Unreliable, point-to-point
- **WS-Routing**
  - Path a message should take
- **WS-Referral**
  - Relaying/delegating service requests
- **WS-Addressing**
- **WS-TransmissionControl**

### Current State

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Discovery &amp; advertising</th>
<th>Description</th>
<th>Services (Tx, routing, …)</th>
<th>Messaging</th>
<th>Data format representation</th>
<th>Data representation</th>
<th>Transport</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Security &amp; privacy</th>
<th>Quality of Service</th>
<th>Management &amp; monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

189


**Immaturity**

- Core standards in place
  - SOAP, UDDI, WSDL
- Competing standards elsewhere
- Standards without wide industry support
  - Very Darwinian standards process
  - (IBM & MS) vs (Sun & Oracle)
- Can connect systems
  - But not much more today

**Vendor Support**

- **Microsoft**
  - Web Services and XML at core of .NET
    - .NET remoting based on SOAP
    - UDDI as local service directory
    - Transparent use of Web Services from Visual Studio (SOAP and WSDL)
    - Early release of proposed standards
- **IBM**
  -Extensive tools support in WebSphere
  - Alphas and betas for proposed standards
Vendor Support

- Sun and Java
  - Started late (dalliance with ebXML)
  - Standard support for Web Services mid-2003 in J2EE 1.4
  - Proprietary support today by app server vendors

Performance?

![J2EE Throughput Graph](image-url)
Back to the Vision

• Vision is about rich, seamless interoperability
  – My client application talking to your server application, regardless of platforms, languages, …

• Not a reality today but close
  – Pretty good for simple tasks
  – Other standards too immature
  – Critical role for WS-Interop

WS-Interop

• Industry body formed to ensure interoperability vision is achieved
  – Standard profiles of current standards
    • Simplifying & ensuring common subsets
  – Compatibility testing
  – All major vendors participating
    • Eventually even Sun… after some politics
  – But WS is getting to be a big set of standards…
    • Will they all survive?
Conclusion

• Is Web Services fact or fable?
  – It’s a good story – and the only one around
  – Major industry investments should make the technology work
  – Huge returns today for customers in making application integration easier
    • And everyone needs application integration
  – Potential for a new Web and new kinds of applications

References

• Microsoft & IBM Web sites
  – http://msdn.microsoft.com/webservices

• Especially GXA papers/specifications
  – Global XML Web Services Architecture
    • Roadmap for WS development
Part VII

Business Process Management

Sub-Topics Outline

• Technologies & requirements for an integration backbone
• Services, service-based architectures, WS
• Programming change towards composite applications
• Key components
• Road ahead: Orchestration in BPM Servers
Key Technologies in an Integration Backbone

- Integration: Orchestration Servers (e.g., Sonic ESB)
- Process Management and Workflow software
  - e.g. Sonic BPM, BEA Process Integrator, MQSI ..
- B2B integration software
  - e.g. Sonic BPM, webMethods, Microsoft ..
- MOM, EAI messaging and adapter technologies
  - e.g. Sonic MQ, MQSeries/MQSI, iWAY, TIBCO ..
- J2EE-compliant application servers
  - e.g. IBM Websphere, BEA Weblogic, Sun iPlanet ..
- Database management software
  - e.g. Oracle, IBM, Microsoft, Sonic XML Server ..
- Enterprise Applications
  - SAP, Siebel, Oracle, ERPs ..

Requirement for an Enterprise–Wide Infrastructure
Sub-Topics Outline

- Technologies & requirements for an integration backbone
- Services, service-based architectures, WS
- Programming change towards composite applications
- Key components
- Road ahead: Orchestration in BPM Servers

Web Services

XML – meta language
SOAP – access protocol
WSDL - description
UDDI - discovery

Not B2B suitable
WS-Security ongoing
WS-Choreography started
Synchronous
Complex “conversations”

Integration and Business means Long-running transaction

- One or more Participants
- Explicit Beginning and End
- Interact with partners, with other divisions, with other departments

Discrete Document Exchanges

Sequencing Rules

Web-service Choreography

- WS, ebXML, other SOA
  - Interface, Document Types, Transition, Delivery
- BPM
  - Orchestration: Invoking Services
  - Choreography: Messaging sequence for complex conversations
  - Long-running, asynchronous

Orchestration Servers: enabling BPM
combining and sequencing the invocation of several different services
Commitment to ebXML

• OAG, RosettaNet, ebXML converge
  – ebXML MSH defines Reliable Messaging
    • ebXML CPA governs payload
    • SOAP, SSL, XMLDSIG, XMLEnc, HTTP, SMTP
  – ebXML BPSS: Collaborations define Messaging Sequence between partners

• Sonic BPM 2.0
  – ebXML Messaging Adapter (ebMS2.0)
    • No java coding – MSH configured with CPA (2.0)
  – BPM Collaborations modeled after BPSS semantics
    • Support in Sonic Integration Studio (IDE)
    • Enforcement of sequencing rules at runtime

• Committees: MSH TC, BPSS, IIC/NIST, Drummond interop test

---

From a Tactical to an Enterprise Solution

[Diagram showing components of a messaging solution including Application, Business Process Engine, Messaging Service, BPSS Document, and Firewall]
Sub-Topics Outline

• Technologies & requirements for an integration backbone
• Services, service-based architectures, WS
  • Programming change towards composite applications
• Key components
• Road ahead: Orchestration in BPM Servers

Change to Component Programming

• Developer to “corporate user”
  – Reduce C++ and java programming

• Development paradigm based on “plumbing”
  – VB, CORBA, EJB/J2EE, SQL, client/server, COM
  – Standard transports such as HTTP/S, RMI, JMS/MQ, etc.
    Services with a document interface

• Abstraction should be based on business functions not technical components, and programming should be mainly process driven

  Service-Oriented Architecture - Composite Applications
Sub-Topics Outline

- Technologies & requirements for an integration backbone
- Services, service-based architectures, WS
- Programming change towards composite applications
- Key components
- Demo & walk-through
- Road ahead: Orchestration in BPM Servers

The 10 pillars of BPM

1. Unified process automation and workflow model
2. Direct model execution and manipulation
3. State management
4. Time-based exception handling
5. Robust process monitoring and analysis
6. Nested model support
7. Concurrent model support
8. Standards based
9. High scalability
10. High reliability

Tyler McDaniel, EAI Journal, 11/2001
Source: Hurvitz Group
BPM: Accommodating Change

- **Process Flow Engine**
  - Abstraction of Collaborations, Services, Roles
    - Document-based interface
  - Definitions independent of Participant
    - Late binding / Polymorphism
- **Integration Framework - Adapters and Transports**
  - Directory Services
    - Partners, Systems, Users treated the same
    - Profiles of individual participants, Groups and defaulting algorithm
  - Abstraction of 3 layers: Allows for mix of Document-Format, Packaging, Transport
- **Business Document Repository**
  - Document storage, state management
  - Separate Message-, Document-, Control-Flow

Separation of Components, Abstraction → Isolation of change

---

**Process Flow Engine**

- Web-service orchestration modeled as “UML Activity Graphs”
- Executes process flows
  - Include multiple partners, speaking multiple XML dialects, in a single workflow
  - Model “long transactions”
  - Synch / Asynch
  - Definitions and instances in XML
- Vital features:
  - Composite states
  - Exceptions, Undo
  - Actions, Extensibility
  - Conversations
Service Polymorphism

- UML Activity
  - Adapter
    - Worklist
    - ebXML
    - Simple
    - EAI
    - Loopback
  - Transport
    - HTTP
    - SMTP Protocol
    - JMS Protocol
    - EAI Protocol
  - Schema
- External Net
  - User
  - Trading Partner
  - MOM Application
  - SAP

Message Flow, Control Flow, and Business Document Lifecycle

- Receive Order Request
- Check Price
- Approve
- Send PO

- Status
  - OK
  - Line Items
  - Latest, Price

- OAG Purchase Order
- Supplier Info

- Transformation Engine
  - Approve
  - Receive Order Request
  - Check Price
  - Send PO

- Services
  - OK
  - Latest Price
Sub-Topics Outline

- Technologies & requirements for an integration backbone
- Services, service-based architectures, WS
- Programming change towards composite applications
- Key components
  - Road ahead: Orchestration in BPM Servers

Orchestration in BPM Servers

*E.g., Sonic Orchestration Server (ESB)*

- Orchestrates …
  - ESB Services
  - Web-Services
  - Progress AppServer
- Process Flow as a Service on the ESB
  - Developed in Sonic Integration Studio
  - Configured, Deployed, and Managed in Sonic Management Console (SMC)
  - Evolve Adapters into ESB
Modeling, Development, Management

Part VIII

Conclusion
Summary

- XML complements EAI/B2Bi/BPM technology, but future EAI/B2Bi/BPM frameworks still need to address XML limitations.
- XML complements application server technology by conveying structured data in a portable way.
- XML provides modeling capabilities and secure message exchange over the Internet.
- MDA/BPM technology promotes “XML Model-Based Computing.”
- Web services platforms provide a shift from traditional enterprise distributed object computing technology to P2P computing with support for pervasive devices, discovery and intelligent agent capabilities.

Readings

- Readings
  - Building J2EE Applications with IBM WebSphere
    - Chapters on Web Services
  - Microsoft .NET Distributed Applications: Building Application Servers
    - Chapters on Web Services
  - Handouts posted on the course web site
  - Explore EAI/B2Bi/BPM, MDA, and Web Services Environments
  - Read related white papers/documentation on EAI/B2Bi/BPM, MDA, and Web Services environments
Project Frameworks

- Project Frameworks Setup (ongoing)
  - Apache Web Server (version 1.3.28/2.2.3+, www.apache.org)
  - Perl (version 5.9.4+, www.perl.com)
  - Microsoft IIS with COM+/.Net and ASP
  - Apache Tomcat
  - (Macromedia JRun4)
  - Apache Cocoon 2/XSP
  - Visibroker, Orbacus
  - RMI-IIOP
  - WebLogic, WebSphere, JBoss
  - ExoLab.org OpenCCM
  - Apache’s XercesJ, XalanJ, XMLSpy, Antenna House XML Formatter, Apache’s FOP, X-smiles
  - JWS, XMI Toolkit, IBM’s Web Service Toolkit
  - POSE, KVM for J2ME, NanoXML
  - IBM (Alphaworks) WSTK, WebMethods, etc.

Assignment

- Explore the references to .Net, Channel Independent, and EAI/B2Bi/BPM Application Server technology
  - Homework #1-4 final due date is 12/07/09
Next Session:
Service Oriented Architectures
Web Services Platforms (Part II)

- Mainstream UDDI Registries and Browsers
- Mainstream Services Toolkits