Application Servers
G22.3033-011

Session 11 - Main Theme
XML-Based Capabilities
in Component-Based Environments

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

- .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
- Assignment #8
Summary of Previous Session

- Web-Enabling Facilities (continued)
  - Java Web Start (JWS)
- COM / DCOM Component Model and OLE
- COM+ MTS Services: Transactions and Security
- COM+ Message Queuing
- .Net/COM+ Integrated Services
- Enterprise Component Development with .Net/COM+
- Designing .Net/COM+ Component-Based Architectures
- Summary
- Readings
- Assignment #7 (continued)

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
- OMAs + Component Models -> J2EE, CCM, DNA
- MDAs with Web Services- and Channels/B2Bi-Enabling services
Part I

.Net Environment
(continued)


.Net CLR Interoperability Support

- **Object binding**
  - Both early and late bound interfaces are supported.

- **Data marshaling and translation**
  - Data type conversion is handled between managed and unmanaged data types.

- **Object lifetime management**
  - Object references are managed to ensure that objects are either released or marked for garbage collection.

- **Object identity**
  - COM object identity rules are enforced.

- **Exception and error handling**
  - The runtime translates COM HRESULT values to .NET exceptions and vice versa
RCW: Runtime Callable Wrapper

The RCW is a managed object and is allocated from the heap maintained by the CLR. As with any other managed object, references to the RCW are traced by the runtime, and the RCW is subject to garbage collection. The RCW is responsible for the following interactions:

- Creating and binding to the underlying COM object.
- Consuming COM interfaces and factoring the interfaces into a managed form.
- Translating and marshaling data between environments.
- Managing the lifetime of the wrapped COM object.
- Translating COM HRESULT values into .NET exceptions.
Calling a .Net Component from COM

CCW: COM Callable Wrapper

The runtime reads the type information for the COM component from its assembly metadata and generates a compatible CCW. The CCW is responsible for the following interactions:

- Creating and binding to the underlying managed object.
- Synthesizing several important COM interfaces (such as IUnknown and IDispatch) based on the object's type information.
- Marshaling and translating data between environments.
- Managing the lifetime of the .NET component.
- Translating .NET exceptions into COM HRESULT values.
COM+ Services

.Net components that participate in COM+ applications are called Serviced Components.
- Serviced Components must be registered in the COM+ catalog using the regsvcs tool (.Net Framework SDK)
- .Net components may be annotated using service-related attributes to specify the exact service requirements
- Serviced Components may share context, transactions, synchronization boundaries with COM+ components.

Calling Native Win32 DLLs

.Net platform supports calling unmanaged coded in native Win32 DLLs.
- Interoperability mechanism is called Platform Invocation (P/Invoke)
- API has to be declared to the .Net runtime for each language
- .Net platform handles marshalling of data types, finds and invokes the correct function in the DLL, and manages the transition from managed to unmanaged code
- CLR also supports callbacks from API functions
- .Net does not currently support calling from a Win32 DLL into .Net managed code (must use COM interoperability then)
Web-Tier Horizontal Migration

(continued)

- Must translate ADO recordsets returned from the middle tier to ADO .NET datasets required by ASP .NET code, typically for data binding.
- May want to enable the use of role-based security between an ASP .NET front end and a COM middle tier by properly configuring impersonation in the ASP .NET application.
- Need to be aware of performance issues when communicating with STA-based COM components from managed code. .NET does not use COM apartments natively and joins a COM MTA by default when interacting with COM. This results in the intervention of a thread-switching proxy.
- Must consider the interoperability and translation of managed and unmanaged data types.
- Must deploy generated interoperability assemblies for middle tier COM components.
- Must deploy the CLR on all Web servers.
To transparently replace middle tier components with .NET components without affecting client code, you will need to maintain the original GUIDS and/or ProgIds of your COM components.

When attempting to transparently replace a COM component, you must properly handle replacement of the class interface generated by Visual Basic components.

You will need to translate the ADO .NET datasets returned from your migrated middle-tier components to ADO recordsets used in your original ASP code.

You must deploy the interoperability assemblies for the middle tier components.
Standard vs. Custom Managed Wrappers

.Net PetShop Logical Architecture
.Net PetShop Physical Deployment Diagram

Internet traffic load is shared either through NLB or similar load balancing technology.

Server A running IIS
- Presentation Tier: ASP.NET Web Forms
- Business Logic Tier: C# Business Components
- Data Access Layer: ADO.NET

Server B running IIS
- Presentation Tier: ASP.NET Web Forms
- Business Logic Tier: C# Business Components
- Data Access Layer: ADO.NET

Account & Orders

ADO.NET

Orders

.Net PetShop 3.0 Application Architecture

Presentation Tier
- ASP.NET Web Forms
- User Interface Processing

Business Logic Tier
- Business Components
- Enterprise Services

Data Access Layer
- DAL Interface
- DAL Factory

Oracle DAAB
Oracle DAL
SQL Server DAL
SQL DAAB
Part II

Connected Devices

*Also See Session 6 Handout on:*
“The Basics of the Palm Platform and the PalmOS Emulator”

---

XML Tools for Information Appliances

- Java Soap library
  - KVM kSOAP ([http://ksoap.enhydra.org/](http://ksoap.enhydra.org/))
- Environment Needed to Run XML-Based Java Applications
  - **Java KVM for J2ME**
    - Java Kilobyte Virtual Machine (KVM) designed to operate with as little as 160 to 512 KB of memory
  - **NanoXML**
    - Lightweight DOM-style XML parser and document generator
  - **Palm OS Emulator (POSE)**
    - Software that emulates the hardware of different Palm devices (Palm III, Vx, VII, etc.)
    - Can be used as a unit testing platform
- Also See [http://webdev.apl.jhu.edu/~rbe/kvm/](http://webdev.apl.jhu.edu/~rbe/kvm/)
Sample Palm Application

- Beaming contact information from a custom Palm database

Part III

XML in Component-Based Environment:
(1) XML and JavaBeans/EJBs
(2) Deployment Descriptors for OMA Services
(3) More on the Persistence Service
(focused on EJBs as CCM is nascent)

Also See Session 6 Handouts on:
“Enterprise JavaBeans Patterns”
“Persistence in EJB Frameworks”
“Efficient CMP Development”
XML Support for JavaBeans and EJBs

- XML and JavaBeans integration
  - e.g., BML, Coins, etc.
- JavaBeans and EJB state persistence using XML
  - XML serialization for Java (e.g., Koala, etc.)
- XML Application Services as JavaBeans/EJBs
  - e.g., Parsing, rendering, querying, messaging beans

Towards XML Model Based Computing

- 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)
Entity Beans in EJB Application Servers

- Represent sets of data (all or part of a database table or a view)
- Functionality limited to creation, update, and deletion of data
- Manage persistence of data
- Maintained in a cache
- Can be container or bean managed
  - Container-managed beans are under the control of an application server for persistence and transaction management
  - Container-managed beans are restricted in the type and complexity of data they can manage
  - Bean-managed beans rely on user provided code for persistence and transaction management
Session Beans
in EJB Application Servers

- Handle the business logic of EJB applications
- May use multiple entity beans to gather application data

Anatomy of a CMP Bean

- EJB remote interface (extends javax.ejb.EJBObject)
  - Contains method signatures for any method that accesses or modifies data stored in the bean
- EJB remote implementation (implements javax.ejb.EntityBean)
  - Provides an implementation of all the methods defined in the remote interface, in addition to methods required by the application server
- EJB home interface (extends javax.ejb.EJBHome interface)
  - Declares method signatures for any method that creates new instances of the bean, and for all methods that are used to retrieve instances of the bean (finder methods)
- EJB key
  - Contains the unique primary key implementation for the bean
  - any class that is a legal value type in RMI-IIOP
Anatomy of a CMP Bean
(continued)

- EJB finder helper interface (optional)
  - Contains one static java.lang.String field for each finder method declared in the EJB home interface
  - Strings are initialized with SQL queries executed dynamically when bean instances are retrieved in a finder method
  - Some server ignore this file and put queries in XML deployment descriptors using proprietary query language formats
- Deployment descriptor (XML -> serialized data object)
  - Names of EJB classes/interfaces, list of persistent fields, etc.
- Database scripts
  - Application server may generate them

EJB Generation
(e.g., IBM EJBMaker, BEA WebGain Studio)

```xml
<?xml version="1.0"?>
<!DOCTYPE ejbmaker SYSTEM "ejbmaker.dtd">
<ejbmaker>
  <!default attributes for beans -->
  <transaction-attr>TX_REQUIRED</transaction-attr>
  <isolation-level>SERIALIZABLE</isolation-level>
  <run-as-mode>SPECIFIED_IDENTITY</run-as-mode>
  <re-entrant>false</re-entrant>

  <!--A Simple Account CMP entity EJB -->
  <bean name = "Account" type = "entity">
    <persistent_field dt="int" col_dt="INTEGER">
      ACCTNUMBER
    </persistent_field>
    <persistent_field dt="java.lang.String" col_dt="VARCHAR(13)">
      SSN
    </persistent_field>
    <persistent_field dt="double" col_dt="DOUBLE">
      BALANCE
    </persistent_field>

    <finder_method name="findNegativeAcct">
      <sql>
        SELECT * FROM EJB.ACCOUNTBEANTBL WHERE BALANCE &lt; 0
      </sql>
    </finder_method>
  </bean>
</ejbmaker>
```
Container Managed Persistence Architecture

![Diagram of Container Managed Persistence Architecture]

Figure 1: Container managed persistence architecture “thin bean/fat container”

Bean Managed Persistence Architecture

![Diagram of Bean Managed Persistence Architecture]

Figure 2: Bean Managed Persistence Architecture “fat bean/thin container”
EJB Development Approach 1

- Handcode Everything:
  - EJB remote interface
  - EJB remote implementation
  - EJB home interface
  - EJB key
  - EJB finder helper interface
  - Deployment descriptor
  - Database scripts

EJB Development Approach #2:
Handcode the Bean and Descriptor
EJB Development Approach #3:
Graphical Bean and Descriptor Generator

Four Patterns for Managing Persistence

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Entity Mgt.</th>
<th>Uses Entity Beans</th>
<th>Uses Session Beans</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Container-managed Entity Beans with Session Beans</td>
<td>Container</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2. Bean-managed Entity Beans with Session Beans</td>
<td>Bean</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3. Entity Beans Only</td>
<td>Container</td>
<td>Yes</td>
<td>No*</td>
</tr>
<tr>
<td>4. Session Beans Only</td>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* Session beans are used in this example for complex database interactions, but they do not serve as a front-end for Entity beans. Instead, they access the database directly as an Entity bean would.
# Container-Managed Entity Beans + Session Beans

<table>
<thead>
<tr>
<th>Entity Management</th>
<th>Uses Entity Beans</th>
<th>Uses Session Beans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

![Diagram](image1)

# Beans-Managed Entity Beans + Session Beans

<table>
<thead>
<tr>
<th>Entity Management</th>
<th>Uses Entity Beans</th>
<th>Uses Session Beans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bean</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

![Diagram](image2)
### Entity Beans Only

<table>
<thead>
<tr>
<th>Entity Management</th>
<th>Uses Entity Beans</th>
<th>Uses Session Beans</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

![Diagram showing Entity Beans Only](image)

### Session Beans Only

<table>
<thead>
<tr>
<th>Entity Management</th>
<th>Uses Entity Beans</th>
<th>Uses Session Beans</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

![Diagram showing Session Beans Only](image)
Performance Characteristics

Pattern Comparison
400 users, 50 iterations, 4 server instances/200MB

<table>
<thead>
<tr>
<th>Pattern Type</th>
<th>Max</th>
<th>Avg</th>
<th>90th</th>
<th>Min</th>
<th>StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Only</td>
<td>6.36</td>
<td>0.20</td>
<td>0.15</td>
<td>0.02</td>
<td>0.612</td>
</tr>
<tr>
<td>Container Managed</td>
<td>6.74</td>
<td>0.31</td>
<td>0.79</td>
<td>0.04</td>
<td>0.796</td>
</tr>
<tr>
<td>Bean Managed</td>
<td>7.02</td>
<td>0.31</td>
<td>0.90</td>
<td>0.04</td>
<td>0.687</td>
</tr>
</tbody>
</table>

Performance Characteristics (continued)

Pattern Comparison
100 users, 50 iterations, 4 server instances/200MB

<table>
<thead>
<tr>
<th>Pattern Type</th>
<th>Max</th>
<th>Avg</th>
<th>90th</th>
<th>Min</th>
<th>StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Only</td>
<td>4.49</td>
<td>0.30</td>
<td>0.35</td>
<td>0.08</td>
<td>0.426</td>
</tr>
<tr>
<td>Container Managed</td>
<td>1.52</td>
<td>0.06</td>
<td>0.07</td>
<td>0.04</td>
<td>0.112</td>
</tr>
</tbody>
</table>
SQL Statements v.s. Finder Methods

<table>
<thead>
<tr>
<th></th>
<th>Max</th>
<th>Avg</th>
<th>90th</th>
<th>Min</th>
<th>StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>5.54</td>
<td>0.20</td>
<td>0.15</td>
<td>0.04</td>
<td>0.571</td>
</tr>
<tr>
<td>Finder Methods</td>
<td>9.99</td>
<td>1.06</td>
<td>2.92</td>
<td>0.05</td>
<td>1.457</td>
</tr>
</tbody>
</table>

Summary of Findings

- **Pattern 1:** (better choice)
  - Great for design and flexibility
    - Good separation of presentation, data, and business logic
  - Fast time to market
- **Pattern 4:**
  - Best for performance and throughput
- **Pattern 3:**
  - Faster time to market than pattern 1
    - No need to code session beans
    - Prohibitive performance cost
Part IV

**EAI, B2Bi, and BPM:**

*Enterprise Application Integration, Business to Business Integration, and Business Process Management*

*Also See Session 11 Presentations:*

“Enterprise Application Integration (EAI)”
“Practical Application of EAI”
“The STP and T+1 Application”

*and the Session 11 Handout on:*

“Enterprise Application Integration (EAI)”

---

**XML & DBMSs Comparison**

- Both separate data from rendition/presentation info
- Similar languages
  - DBMSs: Forms and Reporting, DDL, DQL, DCL
  - XML: XSL, XQL, and processing instructions
- No DML in XML
  - XML is paired with a scripting or programming language
- Validation capabilities
  - DBMSs: datatyping, relationship constraints
  - XML: data type validity and semantic consistency checks
- XML can handle data too complex for some databases
- XML interchangeable form of data vs. multidatabases
MOM Application Development Tools

- Serializing Java objects into XML using reflection
  - Sims Computing lightweight XML messaging framework (based on JMS)
  - xmlBlaster Message Oriented Middleware project
    - MOM platform that uses XML for the message meta-data and quality of service information
    - Messages can be filtered using XPath expressions which match against the XML header document
- Developing MOM applications using the SAX/DOM APIs

Architecture of an XML MOM Application Server

![Architecture of an XML MOM Application Server](image-url)
Sample XML Application Flow

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<author-list>
  <author>
    <au_id>123-456-789</au_id>
    <au_last_name>Smith</au_last_name>
    <au_first_name>Doe</au_first_name>
    <au_email>doe@example.com</au_email>
    <au_phone_number>555-1234</au_phone_number>
    <au_fax_number>555-5678</au_fax_number>
    <au_department>IT</au_department>
    <au_position>Developer</au_position>
  </author>
</author-list>
```

EAI-Based Architectures

- Bas. Process Modeling & Automation
- Transaction Management
- Metadata Management
- Data Transformation
- Reliable Messaging
- Security
- Dev. Tools
- System Administration & Monitoring
- Application Adapters
- Technology Adapters
Comparison of EAI Packages

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Unified Applications Architecture (UAA)</td>
<td>CrossWorlds Software</td>
<td>NS</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>BusinessWare</td>
<td>Vitria Technologies</td>
<td>NSH</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>B2Bi Integration Serv.</td>
<td>WebMethods</td>
<td>NS</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>MQ Series family</td>
<td>IBM/NEON</td>
<td>NSH</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>E*Gate</td>
<td>STC</td>
<td>NS</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Rendezvous</td>
<td>TIBCO</td>
<td>NS</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

H=HP UX, S=Sun Solaris, I=IBM AIX, N=MS Windows, 5 = strongest

B2Bi: B2B commerce and Enterprise Application Integration (EAI)

- B2Bi is based on the transformation and routing of XML documents
- B2Bi patterns:
  - Direct Application Integration
  - Data Exchange
  - Closed Process Integration
  - Open Process Integration
- Existing Frameworks:
  - WebMethods B2Bi EAI framework
  - MQSI (MQSeries Integrator)
- See STP/T+1 in Sub-Topic 2/3 Presentations
**B2Bi Direct Application Integration**

**Architecture**

- Corporation A
  - Application A
  - Application B

- Corporation B
  - Application C
  - Application D

**Integration Broker**

- Adapters
- Transformation

**B2Bi Direct Application Integration**

**Requirements**

- Ability to interact directly with application APIs
- Integration brokers with built-in support for adapters, transformations, and asynchronous content-based routing
- Same Integration Broker on both ends
- Secure transport, component authentication, and user authorizations
- Federated security control
Data Exchange B2Bi

Architecture

Data Exchange B2Bi

Requirements

- Translation of data native to an application into a common document format, and transmission via a gateway
- No constraints on the presence of Integration Brokers
- B2B transactions enabled via a common data exchange format
Closed Process Integration B2Bi

Architecture

Closed Process Integration B2Bi

Requirements

- Principal participant responsible for managing processes
- Other participants are secondary, and do not have visibility into the entire process
- Requires the introduction of business process integration (BPI) services
  - B2Bi product offerings are beginning to incorporate BPI as an essential component
  - In this case, B2Bi enables the integration of logical business process elements expressed as activities rather than data
Open Process Integration B2Bi

Requirements

- Introduces the notion of shared processes rather than operating from a centralized master process manager model
- Each participant is actively managing business processes within its domain
- The BPI layer must support fine-grained control of managed processes
EAI Frameworks and XML

- XML complements EAI technology
  - Powerful meta language
  - Simplicity
  - Separation of content and presentation format
  - Common open standard

- EAI Frameworks must address the limitations of XML
  - Limited Semantics Interpretation
  - Lack of data transformation facilities
  - Inefficiencies of text-based documents
  - Absence of component-based routing

EAI Provides Data Transformations
Efficiency: Binary Objects on the Wire

Part V

Conclusion
Summary

- Java and XML technologies are available on connected devices
- While bean-managed persistence provides the most portability, it generally fall short with respect to productivity.
- Container-managed persistence in EntityBeans provides the most productivity
- Container-managed persistence typically provide the highest degree of performance since, together, the code generation tools and the bean programmer can take advantage of container-specific mapping and caching technology
- Container-managed persistence currently falls short in the area of portability

Summary (continued)

- XML complements application server technology by conveying structured data in a portable way
- XML complements EAI technology, but future EAI framework still need to address XML’s limitations
- 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

- Microsoft .NET Distributed Applications: Building Application Servers
  - Chapters on .Net Remoting and COM+ Services
  - Explore .Net/COM+ Environment
  - Read related white papers/documentation on the .Net/COM+ environments
- Handouts posted on the course web site
- Explore Palm Environment
- Explore EAI/B2Bi Environment
- Read related white papers/documentation on the Palm, and EAI/B2Bi environments

Project Frameworks

- Project Frameworks Setup (ongoing)
  - Apache Web Server (version 1.3.28/2.0.47, www.apache.org)
  - Perl (version 5.8.0, www.perl.com)
  - Microsoft IIS with COM+/.Net and ASP
  - Sun One Active Server Pages 4.0 http://wwws.sun.com/software/chilisoft/index.html
  - Apache Tomcat
  - Macromedia JRun4
  - Apache Cocoon 2/XSP
  - Visibroker, Orbacus
  - RMI-IIOP
  - WebLogic 8.1, WebSphere 5.0, JBoss
  - Inprise AppServer, Sun ONE, Sybase EAServer, Oracle 9i, IONA iPortal, Xoology Concerto, Aligo M-1, Advanced Network Systems Webix
  - GOAL Group OpenCCM, ExoLab.org OpenCCM, icMG K2-CCM (C++), MICO/E (Eiffel ORB), JavaCCM, TAO Group, IONA iPortal (no CCM), Borland AppServer (no CCM), Sourceforge ML-3 ("Mission Impossible 3") and CIF projects
  - 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 #6 due date is 11/12/03
  - Homework #7: due date is 11/26/03
  - Homework #8: due date is 12/10/03

Next Session:
Service Oriented Architectures - Web Services (Part I)

- XML-RPC
- SOAP, DIME, and ROPE
- UDDI, and DISCO
- WSDL