Extreme Java
G22.3033-007

Session 6 - Sub-Topic 1
Distributed Communications Enabling

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Enterprise Web Applications
Basic Web Applications

Client Browser  Web Server  Enterprise Data

Internet Connection  DB

Databases

CGI, SSJS, NSAPI, ISAPI, WAI, Etc.

Robust Web Applications

Presentation Tier  Business Logic Tier  Data Tier

HTTP  Web Servers  Databases

IOP  Documents  Enterprise C/S Applications

Application Server(s)

Rich Clients (Java C/C++, VB)  Databases  Legacy Systems
Application Servers Internals

Client
- HTML
- Java
- C/C++
- Visual Basic

Application Server
- Scalable, Reliable Application Architecture
- Pre-Built System & Application Services
- Distributed Object Infrastructure
- Network Services
- Operating System

- Load Balancing
- Persistence
- Thread Manager
- Fail Safe Services
- Data Access
- Connection Pooling
- Transactions
- Results Caching
- State/Session Mgt
- Directory
- Security
- Messaging

Application Servers Technical Architectures

Open and Extensible

- Extensions
  - TP Monitors
  - Legacy
  - Client/server

- Data Sources
  - RDBMS
  - Oracle
  - Informix
  - Sybase
  - DB2
  - SQL Server
  - OODBMS
  - ODI
  - ODBC & JDBC

- Platforms
  - Sun Solaris
  - HP-UX
  - SGI Irix
  - Windows

- Browsers
- Web Servers
- Open Client Library

- Rich Clients
  - Java, C/C++, VB

- Web Connectors
  - ISAPI, NSAPI, CGI

- Application Server
  - (Java, C/C++)

- Data Access Engine
  - Embedded (bundled) IONA ORB
Application Server Categories

- CORBA-based (OMA-based)
- Java-based (J2EE-based)
- COM+-based
- HTML-Extended
- XML-based
- Adaptive
- http://www.appserver-zone.com/guide.asp lists 77+ unclassified products !!!
- All categories operate on top of a DOC layer

Application Servers Selection Criteria

- Usability
- Scalability
- Security
- Manageability
- Reusability
- Support
- Skills
- The underlying DOC platform is a key driver in the selection process
---

**Distributed Object Computing and Application Servers**

---

**Computing Layers & Facilities**

- Course Outline
  - Distributed Object Computing
  - Application Server Categories
  - Client Interfaces (MOM & POP Applications)
  - Web Application Modeling (UML/XML)
  - Adaptiveness & Model-Driven Application Generation
Distributed Object Computing Models

- CORBA
- EJB & J2EE
- COM+
**Common Object Request Broker Architecture**

- Open distributed computing infrastructure/platform (OMA)
- Programming language independent DOC environment
- Based on OMG IDL, and IIOP/GIOP
- Standardized by the OMG
- Automates common networking programming tasks
  - Object registration, location, activation
  - Request demultiplexing
  - Framing and error handling
  - Parameter marshalling and demarshalling
  - Operation dispatching

**New in Corba 3**

- Component Model (CCM)
- Quality-of-service control
- Messaging invocation model
- Tightened integration with the Internet
- POA (Portable Object Adapter)
- EJB and Java support
- OMG specifications for analysis and design, and application interoperability
  - UML, MOF, XMI, Common Warehouse Model
Overview of Corba:
OMG Reference Model Architecture

Object Services (CORBA services)

- Domain-independent interfaces used by many distributed object programs
- Naming Service: allows clients to find objects based on names
- Trading Service: allows clients to find objects based on their properties
- Other services: persistence, lifecycle management, security, transactions, event notification, etc.
Common Facilities (Horizontal CORBAfacilities)

- Horizontally-oriented interfaces
- Oriented towards end-user applications
- E.g.: Distributed Document Component Facility (DDCF)
  - Compound document facility based on OpenDoc
  - Allows for presentation and interchange of objects based on a document model (e.g., linking of spreadsheet object into a report document)
- Printing, Secure Time, Internationalization, and Mobile Agent Facilities

Domain Interfaces (Domain CORBAfacilities)

- Role similar to Object Services and Common Facilities
- Oriented towards specific application domains
- E.g., Product Data Management (PDM) Enablers for the manufacturing domain
- Other possibilities in the telecommunications, medical, and financial domains
Application Interfaces

- Interfaces developed for a given application
- Not standardized
- Might become candidates for future OMG standardization

CORBA ORB Architecture
**Object Implementation**

- Defines operations that implement a CORBA IDL interface
- Object implementations can be written in a variety of languages
  - C, C++, Java, Smalltalk, Ada, etc.

**Client**

- Program that invokes an operation on an object implementation
- Accessing the services of a remote object is transparent to the caller
  - As simple as calling a method on an object
  - E.g., obj->op(args)
Object Request Broker (ORB)

- Provides mechanism for transparently communicating client requests to target object implementations
- Simplifies distributed programming by decoupling the client from the details of the method invocations
- Client requests appear to be local procedure calls
- ORB is responsible for finding the object implementation, activating it, delivering the request to the object, and returning a response to the caller

ORB Interface

- Abstract interface for an ORB
- ORB may be implemented in various ways
  - One or more processes, set of libraries, etc.
- Interface provides various helper functions
  - Converting object references to strings
  - Creating argument lists for requests made through the dynamic invocation interface
**CORBA IDL Stubs and Skeletons**

- Serve as the “glue” between the client and server applications, and the ORB
- CORBA compiler automates the transformation between CORBA IDL definitions and the target programming language
- Use of compiler reduces the potential for inconsistencies between client stubs and server skeletons
- Use of compiler facilitates automated optimizations

**Dynamic Invocation Interface (DII)**

- Allows a client to directly access the underlying request mechanisms provided by an ORB
- Applications use the DII to dynamically issue requests to objects without requiring IDL interface-specific stubs to be linked in
- DII allows clients to make non-blocking deferred synchronous (separate send and receive operations) and oneway (send-only) calls
Dynamic Skeleton Interface (DSI)

- Server side analogue to the client side DII
- Allows an ORB to deliver requests to an object implementation that does not have compile-time knowledge of the type of the object it is implementing
- Client making the request has no idea whether the implementation is using type-specific IDL skeletons or dynamic skeletons

Object Adapter

- Assists the ORB with delivering requests to the object
- Assists the ORB with activating the object
- Associates object implementation with the ORB
- Can be specialized to provide support for certain object implementation styles (e.g., OODB object adapters for persistence)
ORB Products

- BEA WLE (formerly BEA M3)
- Expertsoft CORBAplus
- IBM WebSphere Product Family (Component Broker)
- Inprise VisiBroker Middleware Products
- IONA Orbix
- Merant CORBA Technology for COBOL
- PeerLogic DAIS
- ObjectSpace Voyager

Questions?
EJB
and
Java 2 Platform Enterprise Edition

Java 2 Platform, Enterprise Edition

- Released on 9/27/99
- Complete development platform architected to meet the needs of enterprise application development
- Makes all Java APIs and functionality available and accessible in a well integrated fashion
- Simplifies the development, deployment, and management of multi-tier server-centric solutions
- Built on Enterprise JavaBeans component architecture
Java Technology Findings

- Open standards are better than closed
- Write Once, Run Anywhere beats proprietary vendor-lock technology
- Findings motivate the creation of a single, unified platform with an easy-to-use model for building distributed ebusiness applications across a wide array of products

J2EE Platform and Enterprise Computing Solutions
J2EE Key Value Propositions

- Simplified enterprise development
  - Supports various pieces of server software
- Industrial strength scalability
- Legacy connectivity
- Open platform (choice/flexibility)
- Security
- Portability

J2EE Platform:
The Whole is Greater than the Sum of its Parts
J2EE: A Complete Computing Environment

- Platform Specification
  - Lists required elements of the platform
  - Lists policies to follow for a valid implementation
- Reference Implementation
  - Semantically correct prototype to test against
- Compatibility Test Suite
  - API-level compatibility, component-level tests, end-to-end compatibility
- Application Programming Model: java.sun.com/j2ee

The Three Cs:
Components, Containers, Connectors
J2EE: Components

- Enterprise JavaBeans
  - Server-side solutions can be built without regards for the database, transaction server, or application they run on
- Servlets
  - Run on vast majority of web servers
- JavaServer Pages
  - Dynamic content leverages off the full power of Java

J2EE: Containers

- Containers provide high-performance, scalable environments for J2EE-enabled servers
- J2EE-enabled servers support EJB-based components, servlets, and JSP-based pages
J2EE: Connectors

- Connectors allow J2EE-based solution to preserve, protect, and leverage off of existing enterprise investments

J2EE: Unifying the Three Cs

- Single platform
- Standard platform-independent technology
- Applications built with components can be run on any J2EE server, and are able to talk to enterprise-class systems that exist today
J2EE Technologies

- JavaIDL
- RMI/IIOP
- JDBC
- JDBC 2.0 Client parts
- JNDI
- Servlets
- JavaServer Pages
- JavaMail, JavaBeans Activation Framework
- JTS, EJB, JTA, JMS, Connector/Resource Mgmt. Comp.
- XML (platform independent data)

Questions?
Goals

Show by example that COM+ makes it:
• As easy to develop Server Components as it is to develop Client Components
• As easy to deliver enterprise apps as it is to deliver workgroup apps!

Technology Scenario

COM+ Services:
- Events
- Security
- Load
- Balancing
- Queued Components
- In Memory
- Database

COM+ The caring, sharing environment for your components
Application Scenario

- Online service like MSN or AOL
- User login causes two databases to be updated
  - Customer Billing Log
  - Customer Login Status
- Total connect time is tracked
- User is notified of various activities while they are logged in (i.e. chat rooms)

Application Flow
Application Object Model

Application Requirements

- Ultimate customer response and availability
- Always accept Logon
- No freebies! Always bill for entire time online
- Use existing billing system
- Use existing dev resources
- Keep billing, Customer Online and other systems consistent
- Real-time two way info on who is on
- Real-time info on what is going on
- Flexible administration
- Maximize use of platform, minimize development time

Many others…
Tackling The Application Requirements

- Ultimate customer response and availability
- Always accept legitimate login
- No freebies! Always bill for entire time online
- Use existing billing system
- Use existing dev resources
- Others…

- Keep billing, Customer Online and other systems consistent
- Real-time two way info on who is on
- Real-time info on what is going on
- Flexible Administration
- Maximize use of platform, minimize development time

Requirement: Simplify Server Side Development

- Issues:
  - Threading
  - Synchronization
  - Data Base Connection pooling
  - Communications - front and back ends
Requirement: Simplify Server Side Development

• Solution: core COM+ Services
  – You write a “single user” component as though it were the only user of DB resources
  – COM+ provides the rich service environment which makes your component multi-user, multi-threaded and resource pooled

Core Application Services

• COM+ provides:
  – Administrator controlled process placement
  – The context
  – Sharing automatically managed by COM+
  – Auto Completion (or explicit SetComplete, SetAbort)

• All you have to do:
  – Stick with the model (or be aware of consequences of leaving model)
  – (Basic component walkthrough)
Tackling The Application Requirements

- Ultimate customer response and availability
- Always accept legitimate login
- No freebies! Always bill for entire time online
- Use existing billing system
- Use existing dev resources
- Others…
- Keep billing, Customer Online and other systems consistent
- Real-time two way info on who is on
- Real-time info on what is going on
- Flexible administration
- Maximize use of platform, minimize development time

Flexible Administration

- Issues
  - Different Administration dimensions
  - Application Distribution
  - Connectivity

Now!

You've Written the App already?

Write an Application to Manage it!
Flexible Administration

- Solution: COM+ Administration
  - Features: extends administration to all COM Components, supports administration of new services and provides a complete programmatic interface for installation, configuration and deployment automation.

Administrative Model

- Applications are defined and components are installed in the applications that they are to run in
- Attributes are set declaratively on the components
  - Transactions
  - Security roles
  - Activation properties
- SDK - complete automation capabilities
Administration Example

- Walkthrough of Billing system deployment
- Billing components are updated on the dev machine
- The Application can be exported manually:
  - Application information and the DLLs to be moved to the accounting server and…
  - ActiveX® Control is downloaded to the client using the CODEBASE tag and a CLSID
  - Or…
- Auto download of components on demand
  - Requires Windows 2000 Directory Service

Tackling The Application Requirements

- Ultimate customer response and availability
- Always accept legitimate login
- No freebies! Always bill for entire time online
- Use existing billing system
- Use existing dev resources
- Other…
- Keep billing, Customer Online and other systems consistent
- Real-time two way info on who is on
- Real-time info on what is going on
- Flexible administration
- Maximize use of platform, minimize development time
Database Consistency

• Issues:
  – Updating multiple Resource Managers
    • (DB, IMDB, MSMQ)
  – Avoiding conflicts
  – Multiple simultaneous users
  – Indeterminate numbers of users

• Solution: COM+ Transaction Services
  – Features:
    • System provided transactions on activation, commit on deactivation, abort on errors
    • System enforced serialization
    • Manual (SetComplete, SetAbort)
      or Auto completion
Consistency Example

- COM+ Transaction Services
  - Transaction code and attribute walkthrough
  - Bus_Customer calls db_CustomerBilling and db_CustOnlineStatus
  - Distributed transaction occurs (transparently) when user logs on
  - Without Transactions customer could be billed for time even if they failed to complete the logon or…
  - They might not get billed even though they successfully logged on
  - Auto Completion (not in the Preview)
  - Transaction code and attribute walkthrough

Tackling The Application Requirements

- Ultimate customer response and availability
- Always accept legitimate logon
- No freebies! Always bill for entire time online
- Use existing billing system
- Use existing dev resources
- Other…

- Keep billing, Customer Online and other systems consistent
- Real-time two way info on who is on
- Real-time info on what is going on
- Flexible administration
- Maximize use of platform, minimize development time
Application Security

• Issues:
  – Security Dimensions
  – Distributed application
  – Components being Compounded

Application Security

• Solution: COM+ Security Services:
  – Role-based access control on classes, interfaces and METHODS,
  – mapping roles to Windows 2000 domain accounts
Security Example

- Application context
  - Customer billing information needs to be protected
  - When a user logs on the bus.Customer component running on the Web server will call the accounting server
  - Using roles the Web server will be allowed to call the accounting server but you may not call the component directly
  - Roles established in package definition
  - Roles established in the IDE
  - Code and Attribute walkthrough

Tackling The Application Requirements - The Story So Far

- Ultimate customer response and availability
- Always accept legitimate logon
- No freebies! Always bill for entire time online
- Use existing billing system
- Use existing dev resources
- Other…
- Keep billing, Customer Online and other systems consistent
- Real-time two way info on who is on
- Real-time info on what is going on
- Flexible administration
- Maximize use of platform, minimize development time
Tackling Other Application Requirements

- Ultimate customer response and availability
- Always accept legitimate logon
- No freebies! Always bill for entire time online
- Use existing billing system
- Use existing dev resources
- Other…

- Keep billing, Customer Online and other systems consistent
- Real-time two way info on who is on
- Real-time info on what is going on
- Flexible administration
- Maximize use of platform, minimize development time

Even Easier!

- Coming soon
  - Direct manipulation of attributes from IDE
  - Direct manipulation of COM+ administration from IDE
- The COM+ Wizard
Conclusion

COM+ makes it:
• As easy to develop Server Components as it is to develop Client Components
• As easy to deliver enterprise apps as it is to deliver workgroup apps!

Call To Action

• Architecting, designing, developing:
  – Assume COM+ level of service and use Windows 2000 now

• Deploying and in production:
  – Use Microsoft Transaction Server now - your components will be uplifted when COM+ is installed

• Everything discussed is a standard feature of Windows 2000
References on COM+

- Books
  - Chappell. *Understanding ActiveX and OLE*
  - Eddon. *Inside DCOM*
  - Grimes. *Professional DCOM*
  - Box. *Essential COM*
- COM Home Page
  - http://www.microsoft.com/com/
- Windows 2000 Platform SDK
  - NT 5.0 Beta 2 or later

Questions?