Distributed Objects and Components

by

Chris Davis

Who am I?

- 4th Year undergraduate
- MSci Computer Science
- You can contact me at:

  c.davis@cs.ucl.ac.uk

Outline

- Motivation
- Objects and components
- Middleware technologies:
  - COM
  - CORBA
  - J2EE
- In-depth: J2EE and Enterprise Java Beans

Motivation

- Main programming languages do not support distributed system construction well
- Local component models do not support interaction across machine boundaries
- Heterogeneity of programming languages

Objects

"has state, behavior, and identity; the structure and behavior of similar objects are defined in their common class"

(Booch, 1994)

"represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain"

(Smith and Tockey)

"a concept, abstraction or thing with crisp boundaries and meaning for the problem at hand"

(Rumbaugh, 1991)

Components

"A self-contained entity that exports functionality to its environment and may also import functionality from its environment using well-defined and open interfaces"

(Stal, 1998)

Distributed Components

- Distributed Components:
  - Utilise communication middleware
  - May exist on separate hosts
  - Across a heterogeneous network
  - Legacy assets may be leveraged
  - Components interoperate as a unified whole

"The Network is the Computer"
(Sun Microsystems)

Objects and Components

More advanced

- Objects
  - Isolated, centrally located
- Distributed Objects
  - Calls between applications
  - Management and performance issues with small remote objects
- Distributed Systems
  - Multi-tier systems, point-to-point connectivity
  - Expensive and hard to develop
- Distributed Components
  - Framework for pluggable components

Any Questions?

Any Questions?

Middleware

- Layer between components
- Provides transparent distribution
- Resolves heterogeneity of:
  - Hardware
  - Operating Systems
  - Programming Languages

Middleware (2)

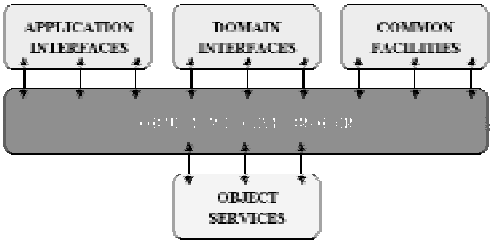
- Transaction-Oriented:
  - BEA Tuxedo
- Message-Oriented:
  - IBM MQSeries
- RPC Systems:
  - Sun RPC
- Object-Oriented:
  - CORBA
  - DCOM
  - J2EE

CORBA – Overview

- Support distributed and heterogeneous object requests
- Transparent to users and programmers
- Facilitate integration of new components into legacy systems

- Defined by OMG
- Open standard
- Used extensively in industry

CORBA – Architecture



CORBA – Architecture (2)

- Application Interfaces
  - Developed specifically for a given application
- Domain Interfaces
  - Interfaces for services for specific domains
- Common Facilities
  - Services targeted to application
- Object Request Brokers
  - communicates requests to object implementations
- Object Services
  - Naming, trading services etc.

CORBA – IDL

- Defines interface to components
- Language-independent
- Compiler generates:
  - Stubs (client)
  - Skeletons (server)
- Stubs and skeletons:
  - Perform marshalling and un-marshalling
  - Resolve heterogeneity between platforms

DCOM – Overview

- Distributed Component Object Model
- Components can be developed:
  - Without need to recompile clients when servers are changed
  - In different environments and languages
- Developed by Microsoft

- Interfaces expressed in Microsoft IDL
- Object implementations in bound language

DCOM – Microsoft IDL

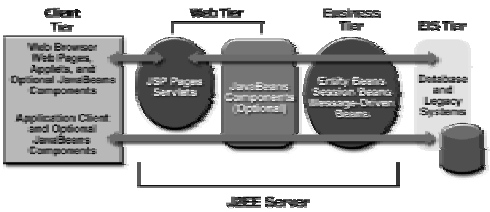
- Contains description of interface between the client and the server programs
- Based on the syntax of the C programming language
- Multiple programming language bindings are available:
  - MS Visual Basic
  - MS Visual C++
  - MS Visual J++
  - and others…

J2EE – Overview

- Java 2 Enterprise Edition
- Multi-tier architecture
- Developed by Sun

- J2EE Components:
  - A self-contained functional software unit
  - Are assembled into a J2EE application with related classes
  - Communicates with other components

J2EE – n-Tier Architecture



Source: Sun Microsystems, J2EE Tutorial

J2EE – n-Tier Architecture (2)

- Client Tier:
  - Web browser based or client application
- Web Tier:
  - JSP/Servlets or Direct communication
- Business Tier:
  - Consists of beans (session, entity, message)
  - Business logic located in reusable components
- Enterprise Information System Tier:
  - database systems
  - enterprise resource planning (ERP)
  - transaction processing

J2EE - Beans

J2EE – EJB

- Enterprise Java Beans:
  - Provide *Business Logic*
  - Exist in middle tier between clients and EIS Tier
  - Consist of Java classes
  - Use RMI or JMS for communication
  - EJB Containers control component execution
  - Standardizes the development and deployment of server components built in Java

J2EE – EJB Session Beans

- Represents single interactive session
- Transient
- Stateful:
  - Hold conversational state
  - One for each client
- Or Stateless:
  - No state held outside calls
  - Beans may be pooled and reused

J2EE – EJB Session Beans – Examples

- Stateless session EJBs:
  - an EJB that calculates sin(x)
  - an EJB that validates a stock symbol x
- Stateful session EJBs:
  - an EJB that books a flight from a form on a website
  - an EJB that orders spare parts for a car as part of an application

J2EE – EJB Entity Beans

- In-memory copy of persistent data
- Represent data
- Persistent
  - Saved to stable when server shuts down
- Allow shared access
  - Multiple clients may read values and update entity beans
- Primary Key
  - ID to enables client to find a specific entity bean
- Relationships
  - May be associated with other entity beans

J2EE – EJB Entity Beans – Examples

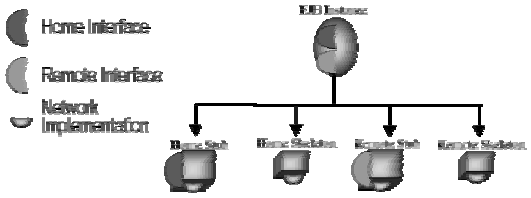
- an EJB that represents a stock's historic prices
- an EJB that represents a genome sequence
- an EJB that represents a footballer player's career statistics
- an EJB that contains your personal profile on a web site

J2EE – EJB Message Beans

- Session beans and entity beans can send JMS messages synchronously
- Stateless
- Asynchronous JMS message consumers
- Uses non-blocking primitive
- Avoid tying up server resources

- Java Message Service
  - Reliable, asynchronous inter-component communication

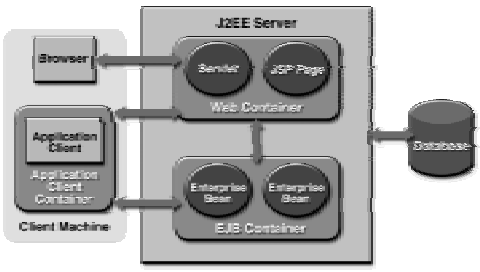
J2EE – EJB Interfaces



J2EE – EJB Interfaces (2)

- Two interfaces clients can use:
  - Home interface
    - Used by clients to create & remove bean
    - Provides meta information
    - Shared among all clients
  - Remote interface
    - Contains business operations

J2EE – EJB Application Servers



J2EE – EJB Containers

- Manages execution of Enterprise Java Beans
- Interface between component and system
- Provide facilities to components:
  - Transaction management
  - Database connection management
  - Security & authentication
  - Remote connectivity
  - Scalability
  - Persistence

Any Questions?

Any Questions?

Summary

- Distributed Components offer many advantages
- CORBA, DCOM and J2EE are in wide use in industry
- CORBA and DCOM have many different language bindings
- EJBs provide easy development, deployment and management of applications
- EJB Containers provide many important facilities for component operation

References

- Engineering Distributed Objects
  Emmerich, W: Wiley and Sons, 2000
- Distributed Systems: Concepts and Design, 3rd Ed
  Coulouris; Dollimore; Kindberg: Addison Wesley, 2001
- The J2EE Tutorial
  http://java.sun.com/j2ee/tutorial/
- Objects and Components
  http://www.cetus-links.org

References (2)

- Your J2EE Community:
  http://www.theserverside.com
- O'Reilly J2EE/Java Community:
  http://www.onjava.com