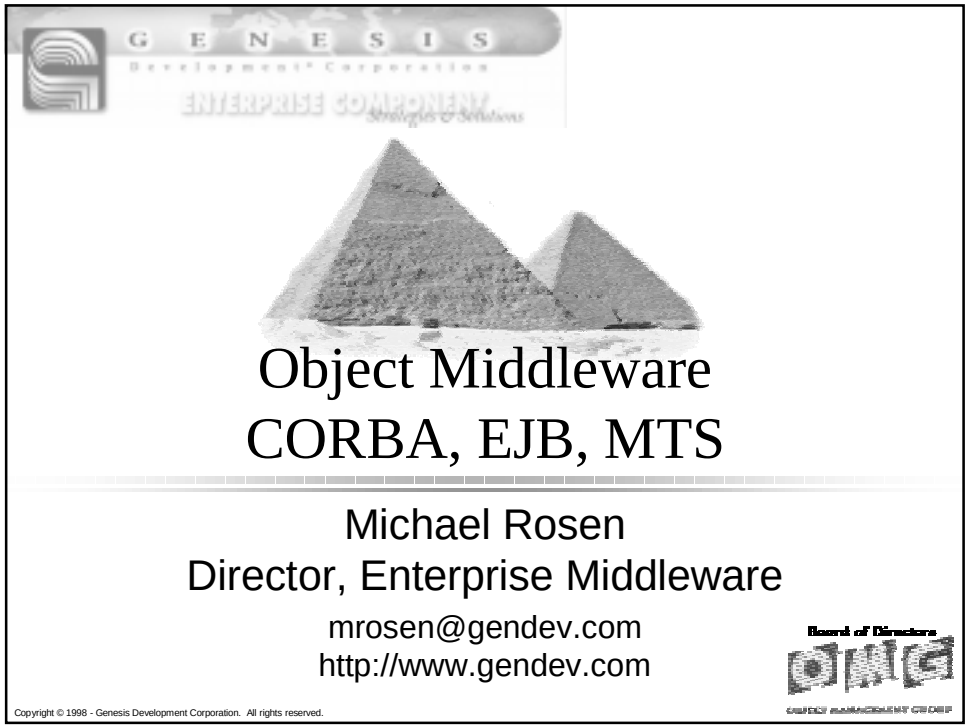


Object Middleware
CORBA, EJB, MTS

Michael Rosen
Director, Enterprise Middleware
mrosen@gendev.com
http://www.gendev.com

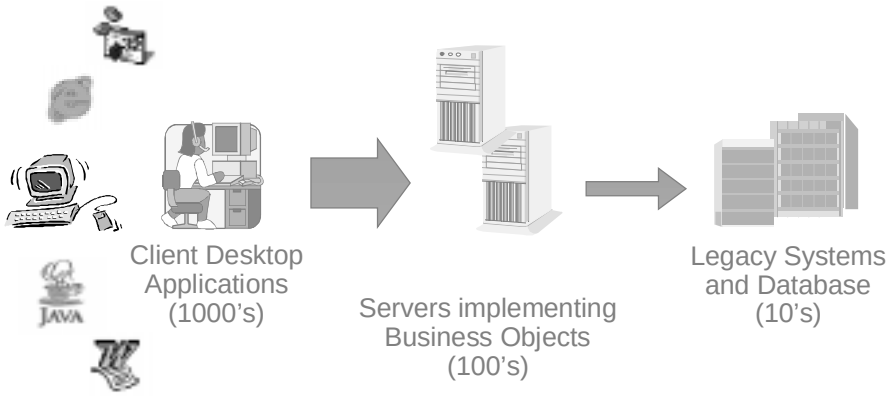
Outline

- Application Trends
- Application Challenges
- Technical Solutions
- CORBA Components
- EJB
- MTS

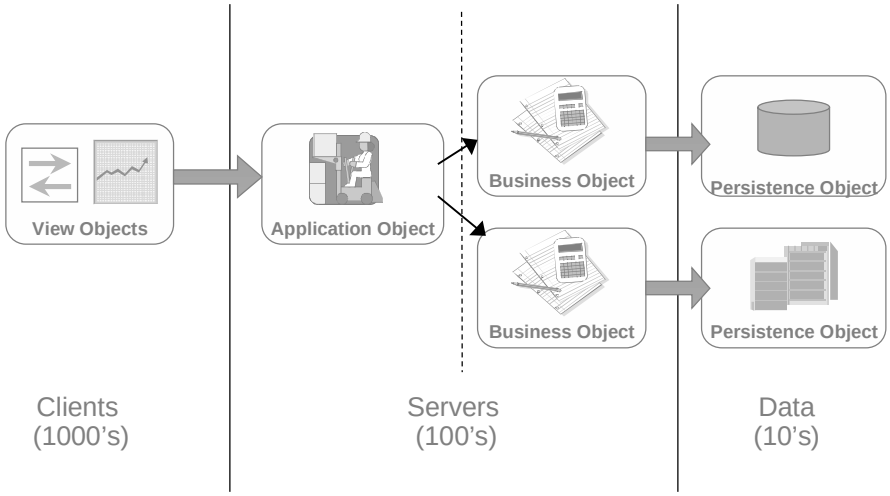
Application Forces

- Enterprise Integration
- Rapid Development
- Constant Change
- Web Enabled
  - Large scale - millions of clients
  - Constant availability - 24 X 365
  - Thin client architecture
  - Aggressive competition

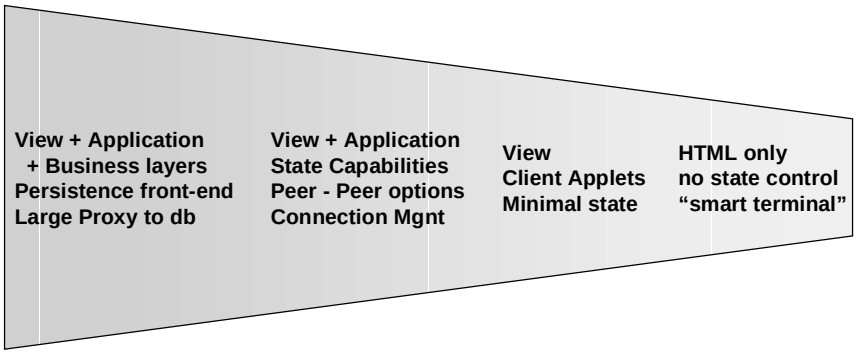
Multi Tier Enterprise Architecture



Multi Layer Application Architecture



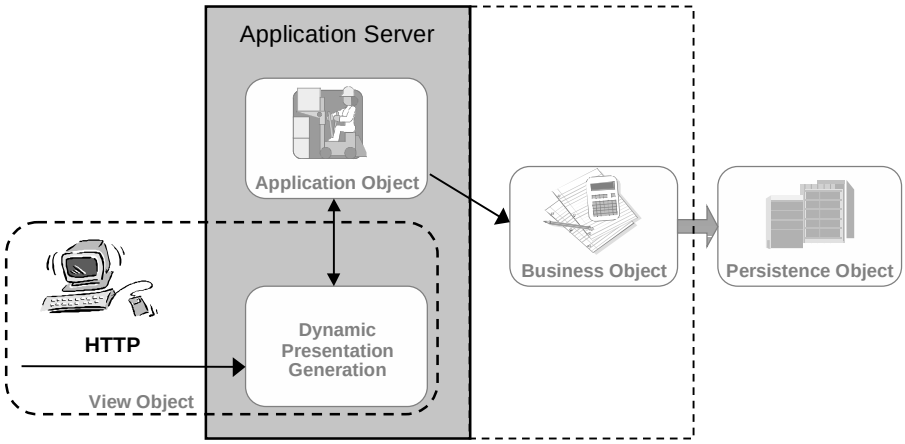
Client Continuum



Server Continuum

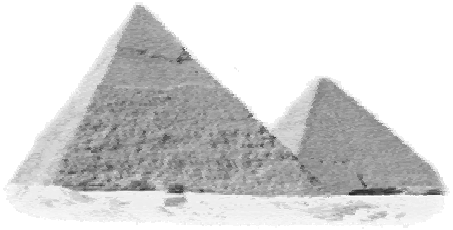
- Legacy and mainframe applications
  - Wrapping and connectivity options
- Procedural/Service applications
  - Integration/Interoperation options
- Custom C, C++, Java applications
- COTS Applications
- Server Components

Application Servers



Infrastructure Support

- A reliable middleware infrastructure will provide features that support large scale applications
  - Component Frameworks
  - State Management
  - Services - Transactions, Security
  - Configuration
  - Management
  - Logging and Recovery



Enterprise Class Middleware

Enterprise Requirements

- Enterprise applications have special Requirements:
  - Openness
  - Interoperability
  - Security
  - Transactions
  - Scalability
  - Configuration
  - Management
  - Availability

Openness/Interoperability

- Many enterprises require an open/standards based solution that provided interoperability with other vendors and legacy applications.
- CORBA provides a true, open solution, but…
- Past CORBA Applications were difficult to build
  - Programming Complexity
  - No system administrative support
  - Must build infrastructure for management
  - Performance concerns
- CORBA Components address this

Security

- Authentication
  - Public Key (RSA), Private Key (Kerberos)
- Authorization
  - Domain, Server, Object, Method
- Encryption
  - Password, message
- Firewalls
- Extremely complicated to implement

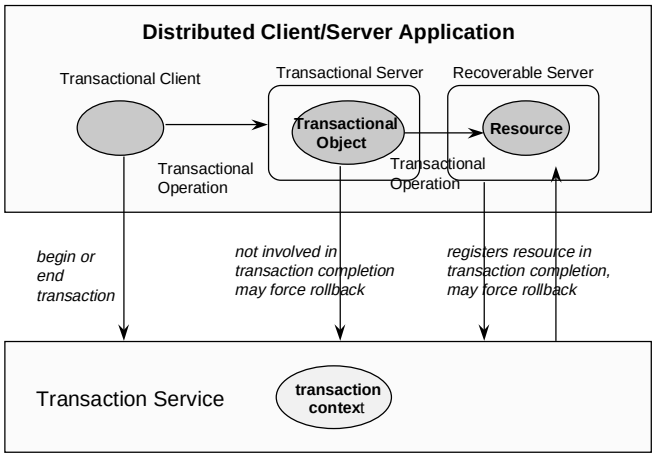
Transactions

- Coordinate changes to multiple resources
- Generic error recovery for distributed applications
- “All or Nothing” outcome
- Resources (DB’s) must have RM interfaces
- System must have a Transaction Manager
- CORBA OTS Interfaces, JTS/JTA Interfaces

Transactions, cont.

- Provide servers with notification of client failure (comm. error or timeout)
- Provide clients with notification of non-transparent server failures
- Prevents loss or inconsistency of data when failures occur during object activation life
- Synchronization provides safe and efficient storage when tied to state management

OTS Architecture



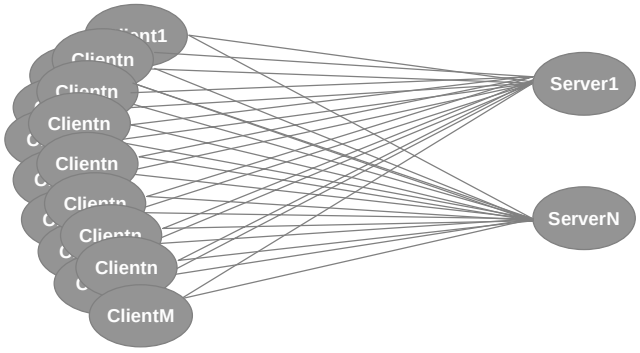
Transaction Synchronization

- New concept introduced for OTS
- *Transactional Object* makes changes to state in memory
- Synchronization interface tells object when transaction is about to *prepare*
- Object can flush state to persistent cache OR veto transaction outcome
- Object now safe to be swapped out of memory (state management)
- State is saved transactionally as part of two phase commit with database

Scalability

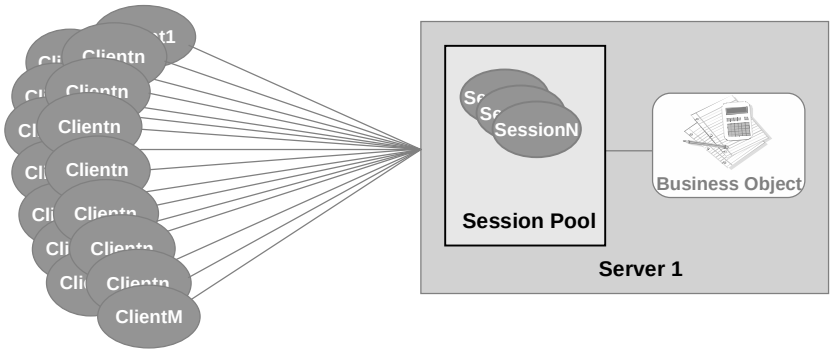
- Scaling reveals resource problems
  - 100's servers
  - 1,000's sessions
  - 10,000's client
  - 1,000,000 objects
- Configuration/Management
- Execution Environment
- Communication concentrator
- Object State Management

Communications Choking



M clients **MxN connections** **N servers**

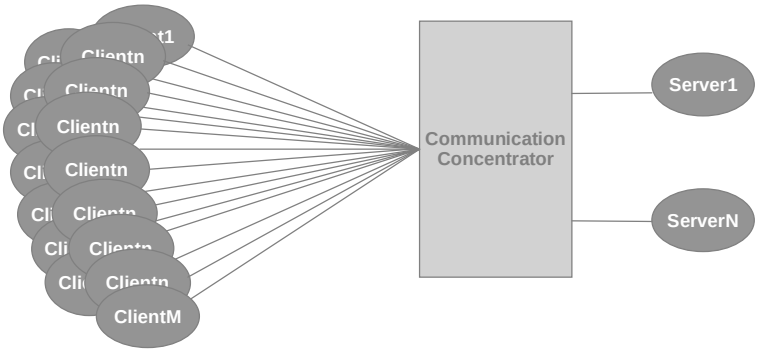
Session Pooling



M clients **MxN connections** **N servers**

(although less server resources consumed)

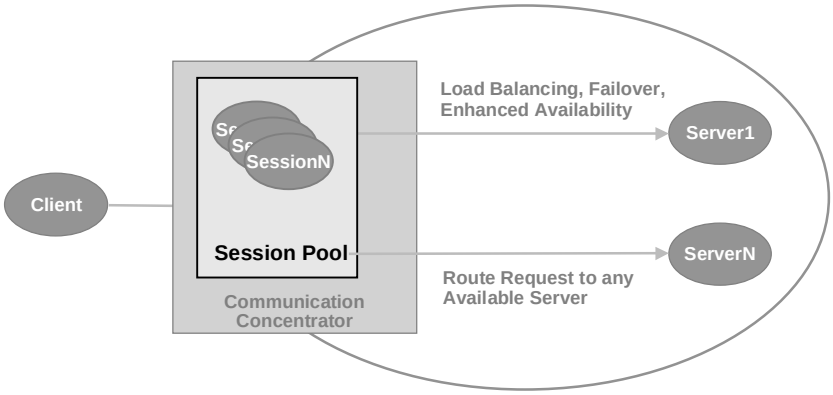
Communications Concentrator



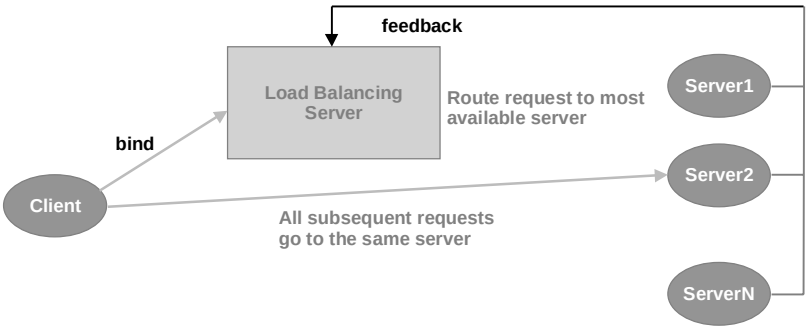
M Clients **Y clients/per concentrator** **N servers**

M \ Y connections

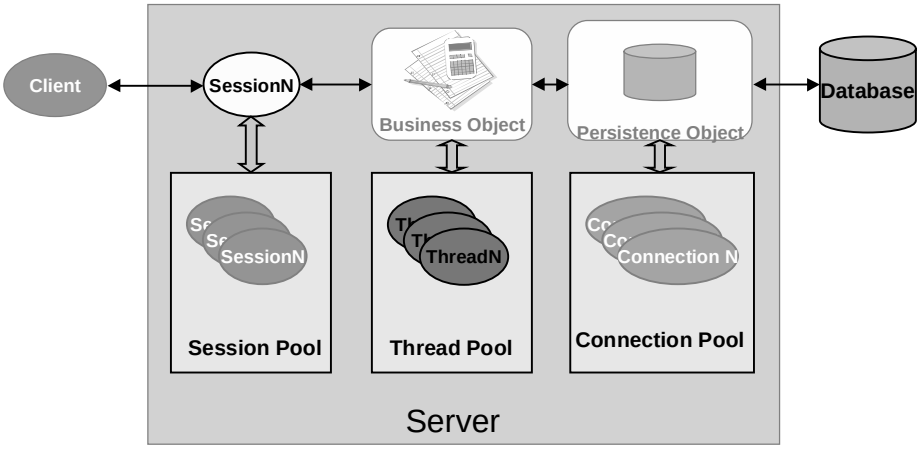
Concentrator Benefits



Load Balancing



Execution Environment



State Management

- Can’t have 1,000,000 objects in memory simultaneously.
- State must be handled by the object implementation
- State management policies are a design-time decision
- Server Frameworks provide help to the implementation

Object State Management

- Allows objects to be swapped into and out of memory between invocations
- An objects state is *restored* when activated in response to a method request
- An objects state is *saved* when the object is swapped out of memory
- Application design describes when and how to manage state
- Server Infrastructure (framework) notifies object at appropriate time

State Management Costs

- Keeping an object in memory
  - resources consumed by the object, e.g. memory, database connections, locks, etc.
  - *relative* to the number of concurrent objects required by the application
- Swapping an object in/out of memory
  - resources consumed to read and write the state to persistent storage
  - *relative* to the access cost of the storage

State Management Models

- Control the *activation life* of the object
- Method Level
  - activation/deactivation on each method
- Transaction Level
  - activation/deactivation on each transaction
- Process Level
  - activation/deactivation on server startup and shutdown
- Application Controlled
  - activation on method, deactivation controlled by app

Pete's Law

"Never have one of anything "

- This principle of scalability is simple.
- The infrastructure requirements to support it are not!

Configuration

- Configuration must be dynamic
- Any single component must be configurable
- Need more clients, add more concentrators
- Need more throughput, add more servers
- Partition application based on request info
- Evolve application with multiple implementations of existing interfaces
- Swap in/out new versions
- All this while the system remains running

Management

- Need to manage different components
  - machines, processes(servers), implementations(services), objects
- Need to manage different things
  - infrastructure and application specific components and statistics
  - event logs
- Need to manage from a single, but movable console
- Integration with SNMP management frameworks

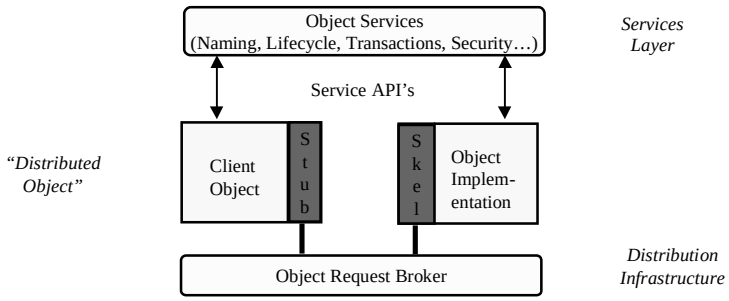
Logging and Recovery

- Detection and automatic restart of servers after failure
  - Load thresholds for startup/shutdown
- Error and event logging
  - System Events
  - Application Events
  - Triggers

Requirement: Availability

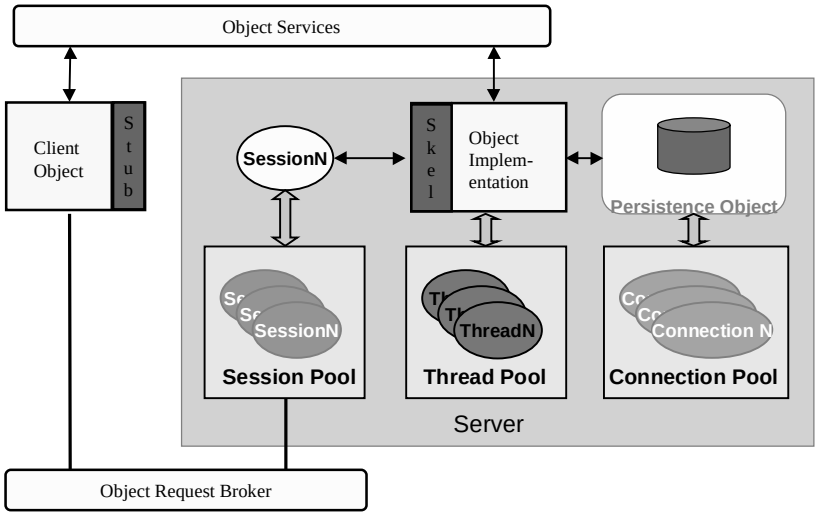
- See Pete's Law
- See Management

ORB Model



- **Client sees distributed object as local, not communication. path**
- **ORB Performs three major functions**
  - Communications and marshalling
  - Creation of remote instance
  - Routing a request from the client to the server

OTM Execution Environment



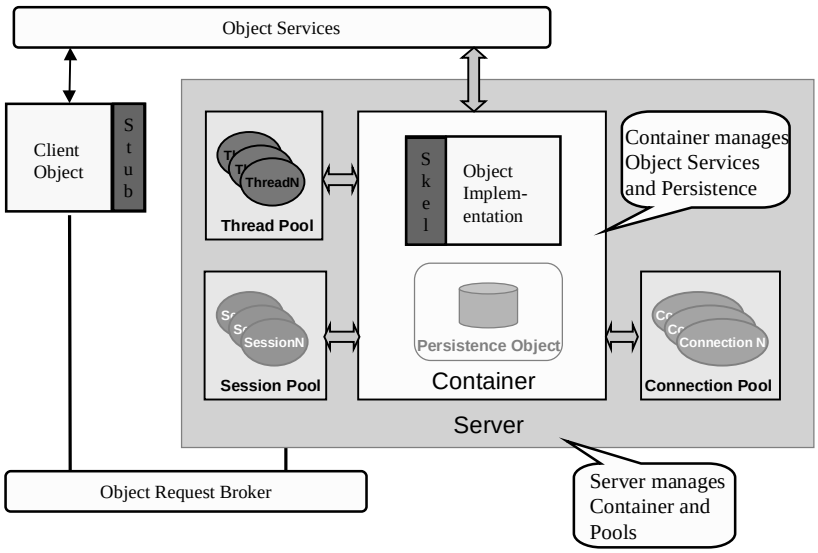
Complex Programming

- Object development is too complicated:
  - multithreading, object registration, security, concurrency control, state management, transactions, persistence, etc.
- Object Services are Complex:
  - Naming, transactions, security, persistence
- Client application is too big:
  - finding object, enrolling in transactions, security, etc.

Server Component Containers

- Dramatically simplifies development
  - only need to write business objects
  - business objects written without complex concerns
- Incorporate object services into server environment
  - transactions
  - security
  - state management
  - persistence
- Provide server component capabilities
- Container is part of an implementation platform

Container Environment



Why Components?

- Time to market
  - Improved application developer productivity
  - Reduced complexity
  - Reuse of existing code
- Programming by assembly (manufacturing) rather than development (engineering)
  - Reduced skills requirements
  - Focus expertise on domain problems
- Key benefit will be seen with server-side development

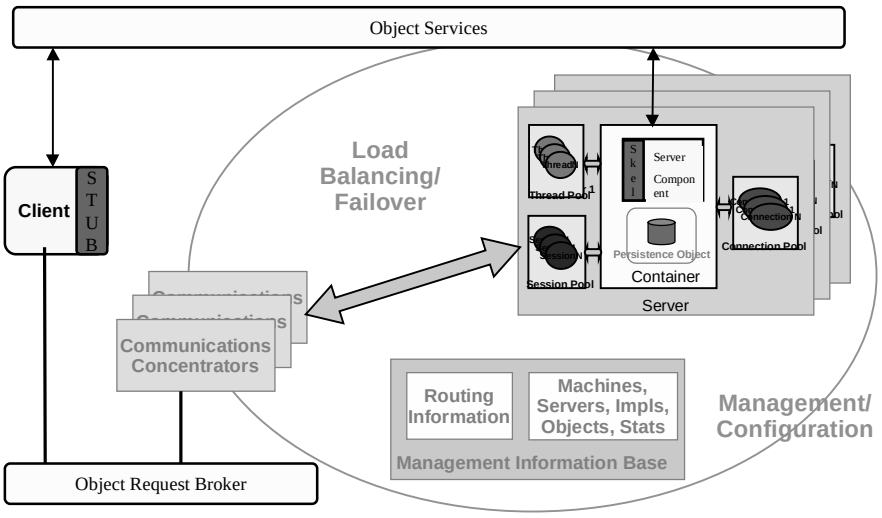
What is a Component?

- A self-contained unit of software code consisting of its own data and logic, with well-defined connections or interfaces exposed for communication.
- It is designed for repeated use in developing applications, either with or without customization.

Contending Technologies

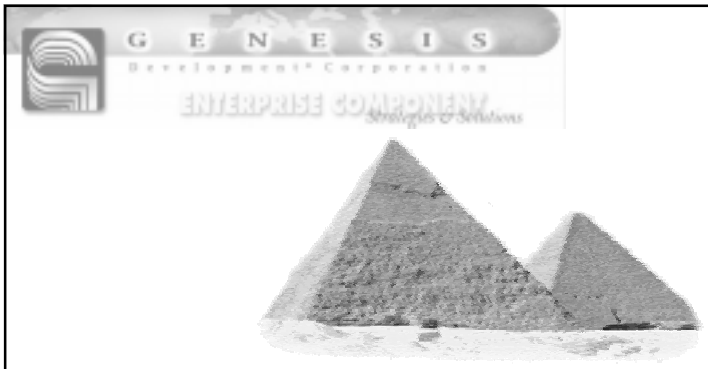
- OTM's
- OMG's CORBA Components
- Enterprise Java Beans (EJB)
- Microsoft's COM +
  Distributed Internet Architecture (DNA)
- All require a robust infrastructure

The Big Picture



OTM's

- Generation of middleware products that combine:
  - Objects: Open, standards based (CORBA, EJB)
  - Transactions: Necessary for data integrity
  - Management: Vital for the enterprise
- All ORB vendors have one. Most OTM's will provide components soon.
- MTS is also an OTM
- "Application Server" is the newest flavor
  - Most aspects of OTMs plus components and Web
  - Typically doesn't have either ORB or TM genealogy



CORBA Components

CORBA Components

- A server-side component model for building and deploying CORBA applications
  - An architecture for defining server components and their interactions
  - A packaging technology for deploying binary, multi-lingual, executables
  - A container framework for injecting security, transactions, events, and persistence
  - A CORBA container for Enterprise JavaBeans

CORBA Component Features

- Support for Multiple interfaces
- Relationships
- Events
- Properties
- Default factories and finders
- Container managed
  - transactions
  - security
  - persistence
  - events

The Container Model

- A Framework for Server Applications
- Built on the Portable Object Adaptor
- Provides Interfaces to component clients
- Supports interfaces for Systems Services
  - transactions
  - security
  - events
  - persistence
- Invokes Callbacks for instance management

Component Categories

- Session
  - model stateless business processes
  - like EJB session beans
- Process
  - model stateful business processes
  - server-managed persistence
- Entity
  - model stateful business entities
  - like EJB entity beans

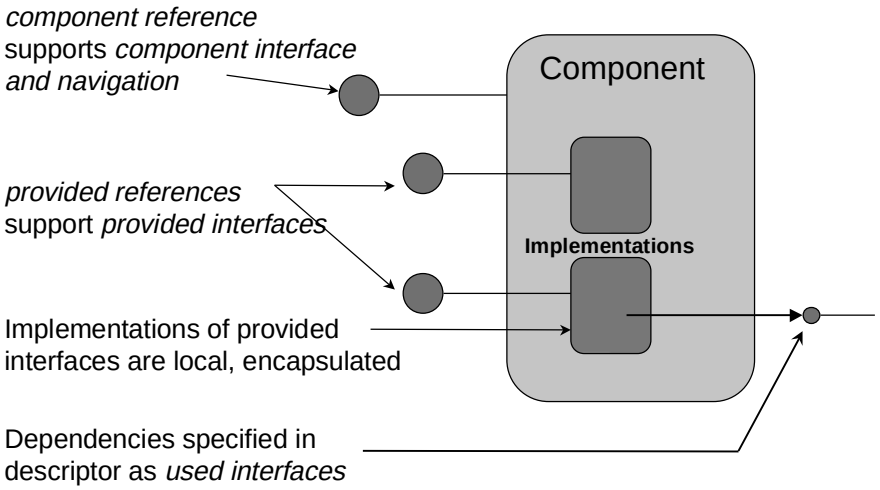
Container Services

- Reference creation
- Instance creation
- Transactions
- Security
- Events
- Persistence
- Servant lifetime management

State Management

- Control the lifetime of the servant
- Four State Management Models
  - Method Level
    - activation/deactivation on each method
  - Transaction Level
    - activation/deactivation on each transaction
  - Process Level
    - activation/deactivation on server startup and shutdown
  - Application Controlled
    - activation on method, deactivation controlled by app
- *Activate* and *Deactivate* allow component to control what is saved/restored

CORBA Component Structure



Events

- Components can emit and consume events
- Simple programming model
  - publish/subscribe
  - structured events
  - push model
  - maps to subset of CORBA notification
- Channel management and quality of service mediated by container
- Interoperability with other CORBA notification implementations

A Day in the Life of a Component

- A component is specified.
- A component is implemented.
- A component is packaged.
- A component may be assembled with other components (usually in a design tool).
- Components and assemblies are deployed.
- The application is a big success...

Roles

Role (human)	Tools	Artifacts
Component Creator	Text editor IDL Compiler IDE Language Compiler Component Packaging Tool	Component Archive File - implementation files - component descriptor - property state file
Application Assembler	Visual Assembly tool (or text editor and packaging tool)	Component Assembly Descriptor - or - Component Assembly Archive - assembly descriptor - component archives
System Deployer	Deployment Tool	Deployed Assembly and Components

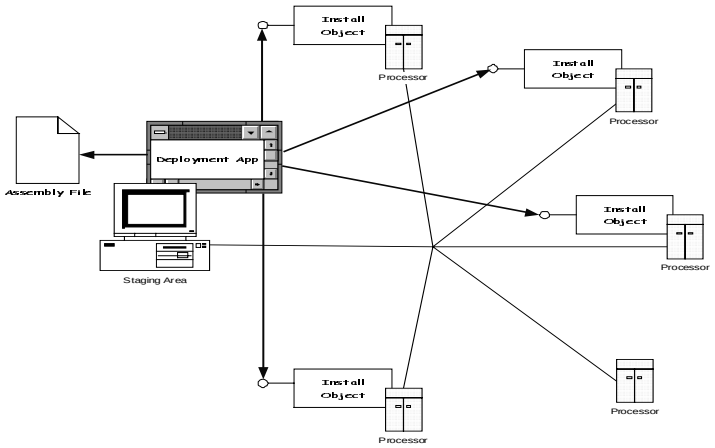
Component Package

- A component package consists of an XML descriptor and a set of files
- The package descriptor contains:
  - one or more implementations of the component
  - pointers to a container descriptor for each implementation and property files
  - general information about the component

Component Assembly

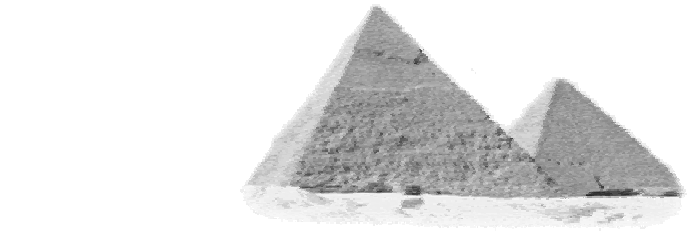
- A component assembly is a pattern or a template for a deployed set of interconnected components.
- Described by a assembly descriptor in terms of component files, partitioning, and connections.
- May be deployed as is or imported into a design tool and reused or extended.

Deployment



Enterprise Java Beans Support

- CORBA components are a proper superset of EJB
- An EJB can be deployed into a CORBA component container
- A Java CORBA component using only EJB equivalent function can be deployed into an EJB container
- Both can be clients of the other
- OMG is working with Sun to define an enhancement process



Enterprise Java Beans

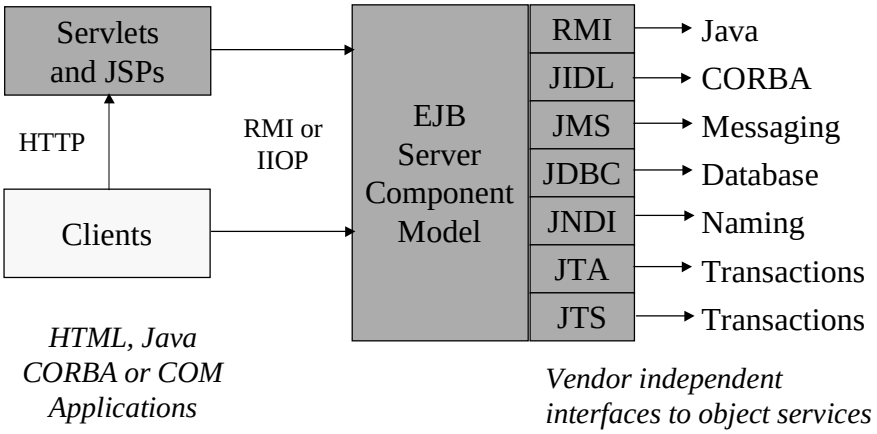
Java on the Server

- Java is becoming a popular choice for server-side programming language
- More productive than C++
  - 25% development
  - 50% testing
- Natively supports distribution
- *Enterprise Java* platform designed to satisfy enterprise requirements

Enterprise Java Platform

- Specifies the minimum set of services and API required for enterprise applications
- One standard, vendor independent interface to any service implementation
- ODBC Metaphor
  - ODBC provides a single interface to any DB
    - Oracle, Sybase, Informix, DB2, etc.
  - JNDI provides a single interface to any naming or directory service
    - DNS, NDS, LDAP, COS Naming, etc.

Enterprise Java Platform



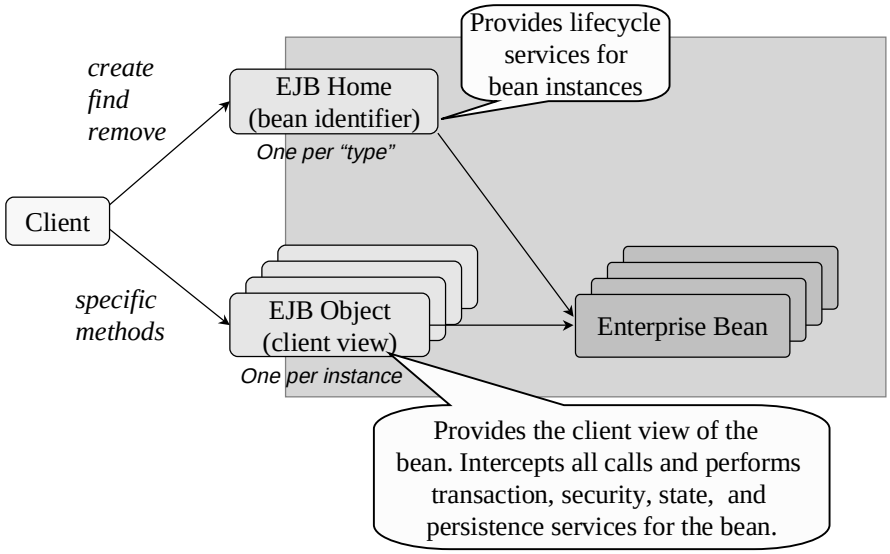
EJB Features

- Simplifies development
- Part of Enterprise Java Platform
- Provides distribution
- Container managed
  - Lifecycle
  - Transactions
  - Security
  - State management
  - Persistence

The Container Model

- A Framework for Server Applications
- Runs inside an EJB Server
  - EJB Specification does not define details of server/container interaction
- Supports interfaces for Object Services
  - transactions, security, peristence
- Provides “wrapper objects” for all beans
  - EJB Home and EJB Object
- Intercepts all calls and automatically provides lifecycle and object services

EJB Container Objects



Container Services

- Registration and lifecycle
- Distribution
- State management
- Persistence
- Transactions
- Security

Component Categories

- Session
  - Model stateless business processes or client work
  - No unique Identifier
  - stateless
    - any instance can be used by any client
    - most scalable approach
  - stateful
    - maintains user state across method calls
    - container automatically manages the state
- Entity
  - Represents data in a database
  - Has a unique identifier (primary key)
  - Can be shared by multiple clients
  - Long duration, recoverable

EJB Persistence

- Bean-Managed Persistence
  - Developer responsible for all code
- Container-Managed Persistence
  - Container responsible for access code
  - Container automatically retrieves objects and stores modifications to/from the database
  - *ejbLoad, ejbStore, ejbRemove* methods allow developer control

EJB State Management

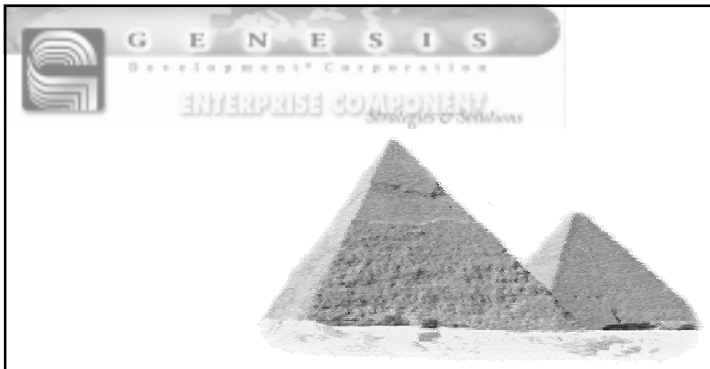
- The container manages state of all EJB instances
  - Automatically saves/restores bean to memory
  - Dependent on the bean category
- *ejbActivate, ejbPassivate* methods
  - Called whenever an instance is to be loaded or removed from memory
- *SessionSynchronization* interface allows bean to synchronize object state with the database

Deployment Descriptor

- Bean development environments provide a tool for building Deployment Descriptors
- Describes installation and runtime attributes about a bean, some can be customized
  - Lifecycle (Class name, Bean Home, Entity versus Session, Environment properties)
  - State management and persistence semantics
  - Transaction semantics
  - Security semantics

Deploying EJB's

- Build the bean
  - Implement the business methods
  - Build deployment descriptor using IDE tool
  - Package in an EJB Jar file
- Deploy using EBJ Server specific tool
  - Install
  - Customize
  - Map Persistence
  - Manage



Microsoft COM+

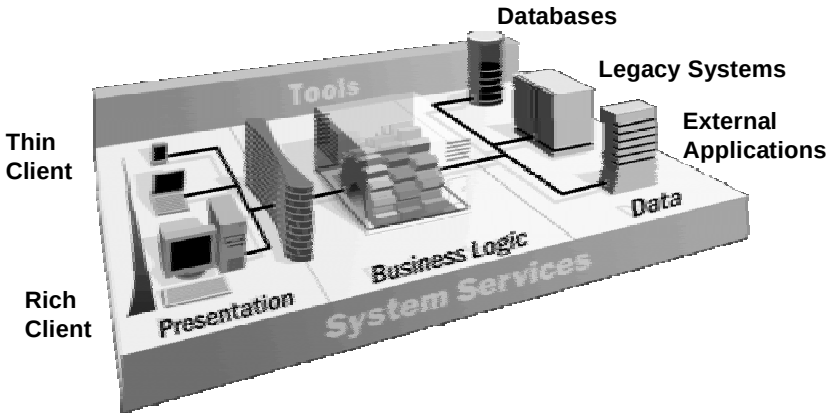
It’s not just MTS

- Microsoft Transaction Server (MTS) is only part of a group of products aimed at enterprise applications
- These products provide a full range of functions from thin clients to databases
- MTS provides an easy to use, component based approach to server development similar to CORBA Components or EJB

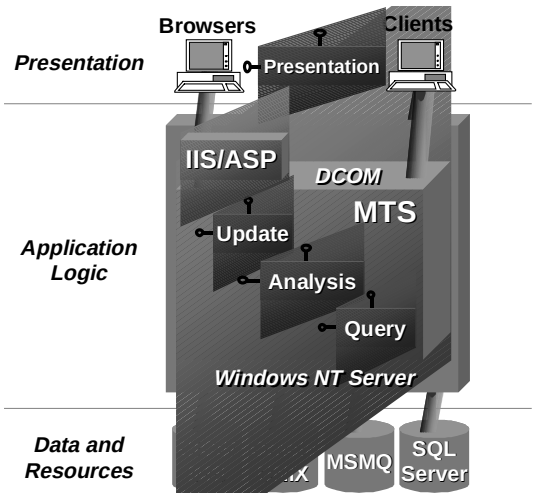
Slide 73

Windows DNA

The development model for Windows



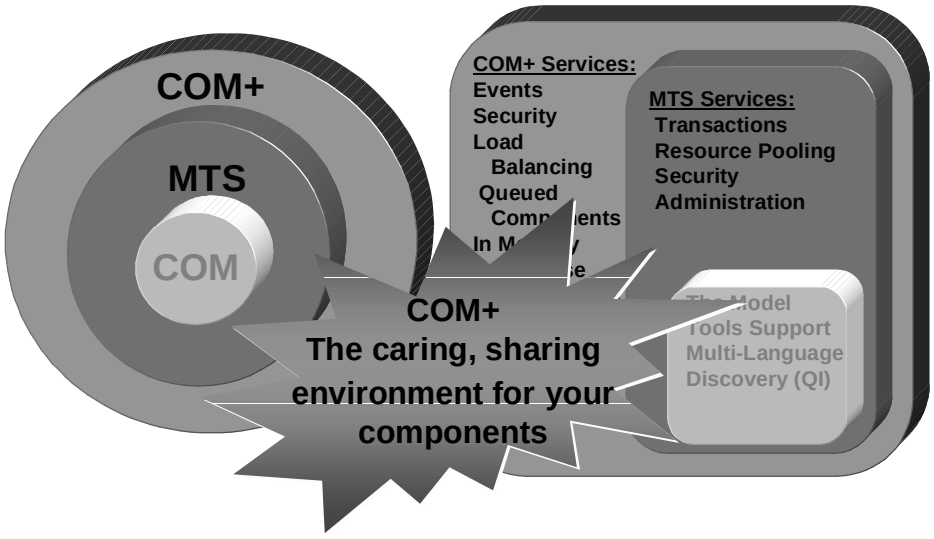
Another View



What Is COM+?

- Unification of COM and MTS
- Plus...
  - Extensions to base COM
  - Extensions to MTS services
  - New Services
- Minimal impact on how you write components
- Remoting architecture doesn't change (DCOM)

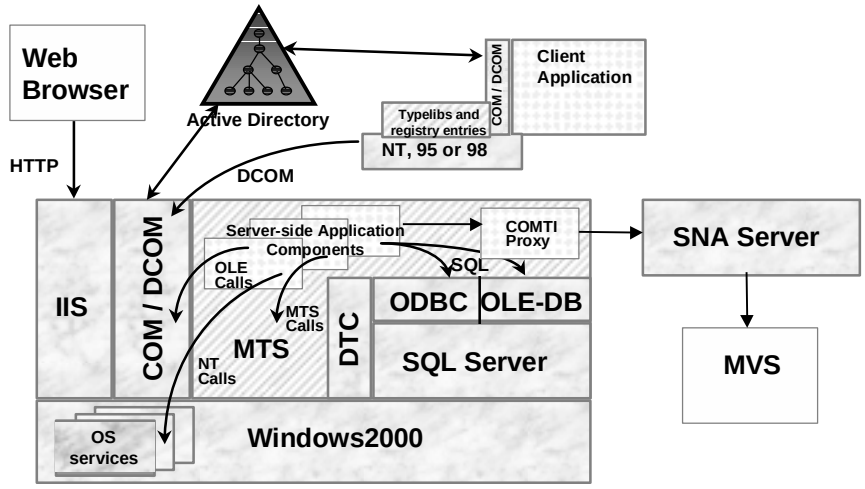
COM+ Technology



COM+ Services

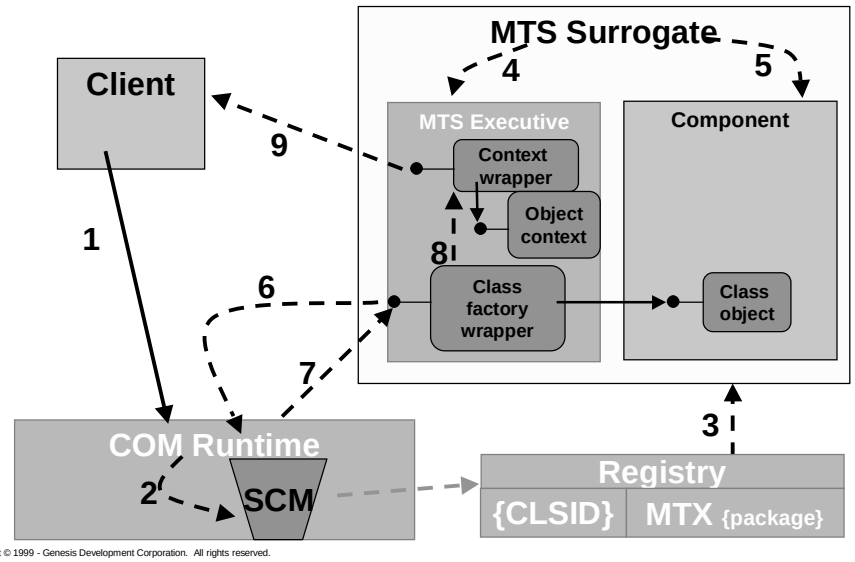
- Build on unified architecture
- Container Services
  - Security
  - Transactions
  - Object Pooling
  - Events
  - Load Balancing
- Other Services
  - Queued Components
  - In-Memory Database
  - Deployment
  - Administration

MTS's Role in COM+

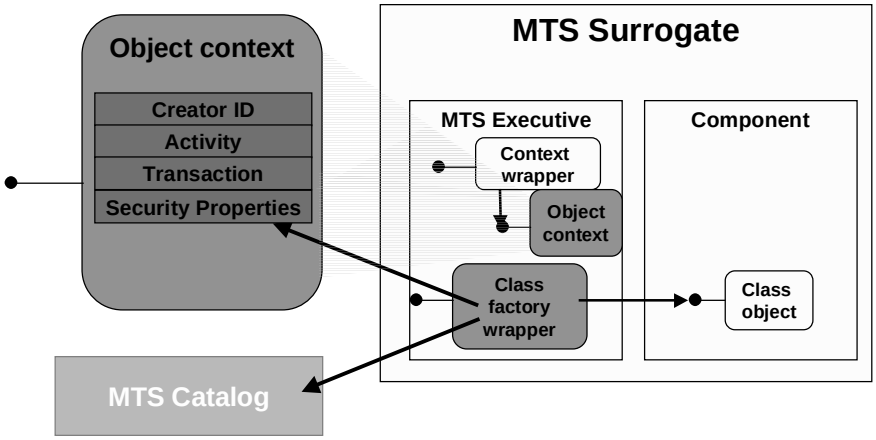


Slide 79

Object Creation w/ MTS



Object Context



MTS Component Responsibilities

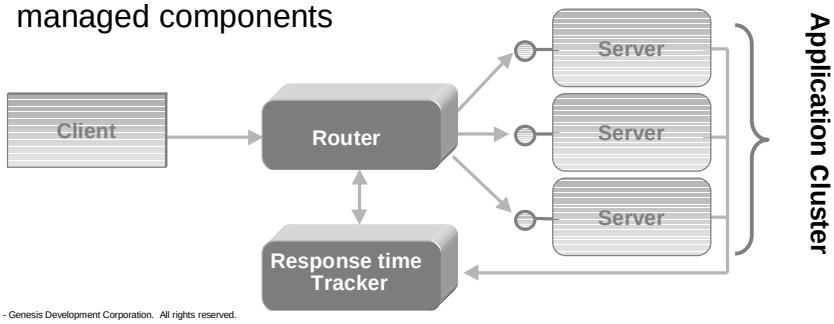
- Component doesn't know about transaction
- Call context.SetComplete when safe to remove object from memory and/or commit transaction
- Recommended to call at end of every method
- No guarantees of behavior

MTS State Management

- No explicit model or policies
- Single setComplete method does both state and transactions
- “One-size-fits-all” model similar to application controlled.
- Recommendation for after every method provides “method-level”

MTS Load Balancing

- Support for COM/MTS components
- Cluster size of up to eight nodes
- Extensible load balancing policy
  - User-pluggable routing mechanism
- Support for multiple simultaneously managed components

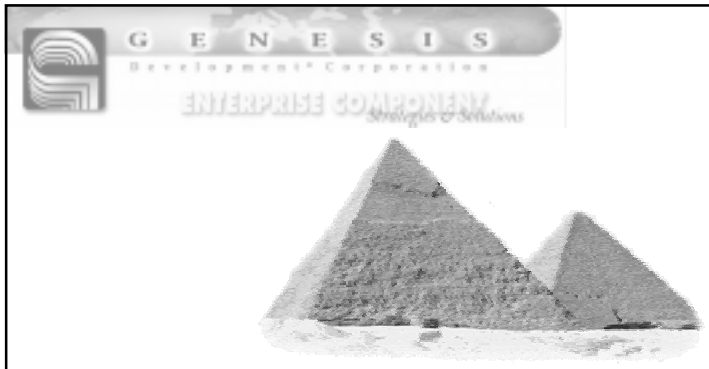


MTS Administration

- MTS Component Administration is done using MTS Explorer
- Scripting interfaces allow automation of (some of) the configuration and deployment
- Server for pushing components/apps to clients.
- Nothing specific for server components

MTS Component Configuration

- Often done by administrator (or customer)
- Take COM+ Server Application
  - .APL, DLLs, TLBs
- Fill in environment specific “details”
  - Role members
  - Specific server names / queues
  - Establish Application Cluster members
  - Etc.
- Re-export the configured application



Comparison Summary

COM Strengths

- Tools for developing user interfaces
- Personal productivity tools
- Integration between desktop applications
- 3rd party client component market
- Ubiquitous Client Platform
- Commodity hardware
- *Trying to bring developer productivity gains to server applications.*

Java Strengths

- Multi-platform, portability
  "write once - run everywhere"
- Inherently supports Web applications
- Excellent integration with CORBA
- Ease of development
- Java Platform for the Enterprise
- Attractive to developers
- *Standards/attitude and performance being addressed.*

CORBA Strengths

- Integration with legacy/heritage systems
- Open, standards based, multi-platform, multi-vendor
- Designed for networking
- Mature, proven, interoperable
- Market segment specific implementations
- CORBA Services
  - naming, security, transactions, etc.
- *Components addressing complexity, productivity issues*

Communication Differences

- CORBA, RMI, and DCOM perform essentially the same functions
- But:
  - Different programming interfaces
  - Different datatypes supported
  - Different languages supported
  - Different wire protocols
  - Some different behaviors

Component Differences

- CORBA Components, EJB, and COM+ perform essentially the same functions
- But:
  - Different Standards (real or imagined)
  - Different programming interfaces
  - Different languages supported
  - Different levels of programmer control
  - Different deployment capabilites

Product Differences

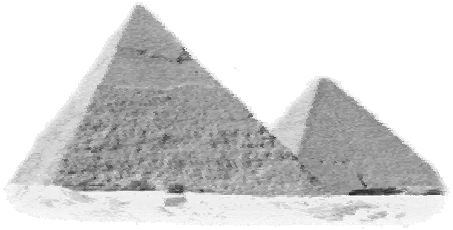
- EJB, CORBA Components, and MTS products target similar markets,
- But:
  - Different target platforms
  - Different in achievable Scale
  - Different Manageability
  - Different Configuration capabilities
  - Different levels of Web integration
  - Different capability for Legacy integration

Comparison Summary

	CORBA Components	**EJB**	**MTS**
Protocol	IIOP	RMI or IIOP	DCOM
Language	C, C++, Java, etc.	Java	C, C++, VB
Interface Specification	CIDL	Java class	MIDL
Interface Definition	CORBA IR	Not specified	Typelib
Memory Management	Full, flexible, 4 models, callbacks	Container managed, callback interfaces	“One-size-fits-all”
Deployment Descriptors	Complex, multiple components	Simple, single component	Simple, single component
Categories	Session, Process, Entity	Session, Entity	Not formalized
Installation	Can specify properties and behaviors	Incomplete	Incomplete
Runtime configuration	Extensive, structural and behavioral	Incomplete	MTS Explorer does some
Metamodel	Formal, MOF based	None	None

Summary

- OTM's and Application Servers attempt to bring "client" productivity gains to server development.
- OTM's likely superceded by App. Servers
- CORBA Components, EJB, MTS are similar component models.
- Products are differentiated by standards, platforms, languages, scale, deployment and management capabilities.



Genesis Development Corp.

10 North Church St.
West Chester, PA 19380
http://www.gendev.com

Distribution Options

- RMI - Java Remote Method Invocation
  - Java to Java communication
- JavaIDL - Java Interface Definition Language
  - CORBA IDL interfaces and IIOP
- JMS - Java Messaging Services
  - Asynchronous communications using message queuing and publish/subscribe

Object Services

- JNDI - Java Naming and Directory Interface
  - Object registration and location
- JTA - Java Transaction API
  - Simplified API for transaction demarcation
- JTS - Java Transaction Service
  - Full Java binding to CORBA OTS for distributed transaction coordination

Database Access Options

- JDBC - Java Database Connectivity
  - Based on ODBC - relational databases
- SQLJ - Embedded SQL (ANSI standard)
  - Java stored procedures
- ODMG Java bindings (ODMG standard)
  - Access to ODBMS
  - Object/relational mapping