Session: Distributed Objects- CORBA, DCOM/SOAP

- Distributed Object Technologies
- CORBA Overview
- CORBA Developments
- DCOM
- Web Services and SOAP

Amjad Umar

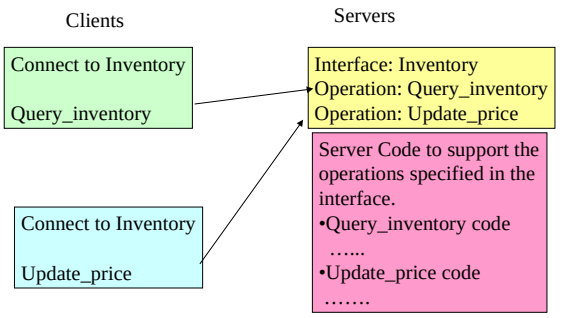
Distributed OBJECT SYSTEMS

- View the world in terms of objects ("things")
- Easier to think of systems as objects (customer, shipping clerk,,)
- main driver: reusability
- Objects are rarely at one site
- Distributed objects allows coomincation between remotely located objects

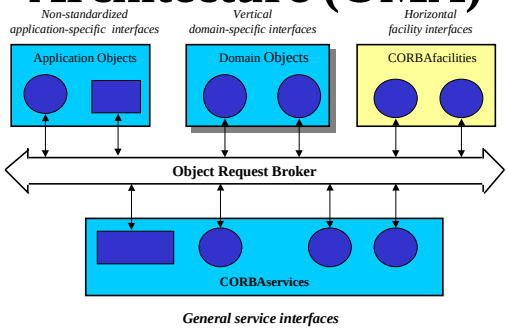
Trends:

- Business objects
- Components (VB, JB, EJB)

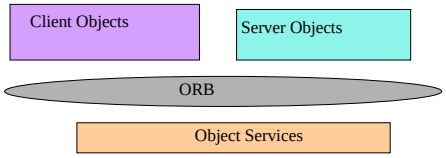
These objects are spread over multiple sites



Object Management Architecture (OMA)

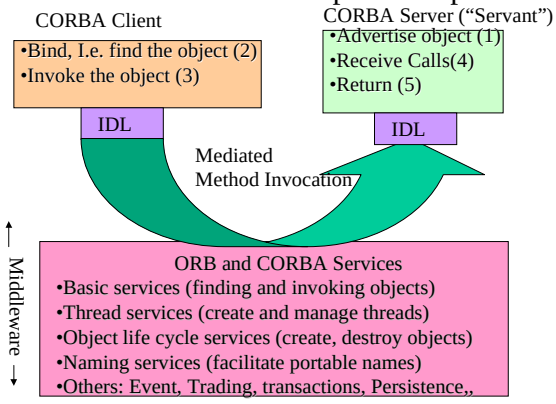


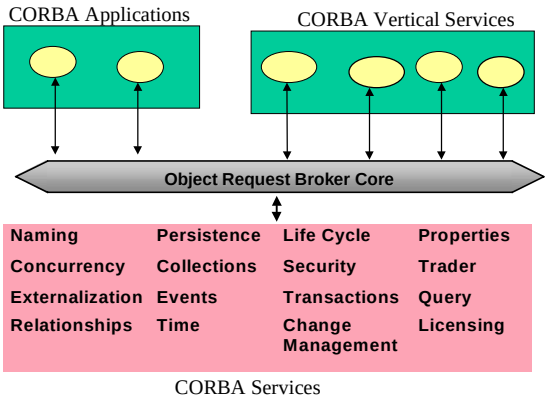
CORBA Facilities



- Application Objects: Application logic in C++, Java, Smalltalk
- Object-Request Brokers (ORB)
  - Support delivery of client requests to distributed objects.
  - ORBs format requests for sending to each object.
  - ORBs support synchronous and asynchronous
- Object Services: Keep track of info about each object, including address of each object in system.
- Common Facilities: For different domains

CORBA At Work: A Simple Example





Expected benefits of the ORB approach

- Provides universal notation for interfaces: IDL;
- Simplifies distributed computing
  - remote and local object invocation are indistiguishable (?)
  - location transparency
  - standard services (registration,…)
  - standard protocol over several transport layers
- Promotes component-based development
  - components are reusable
  - components are designed for integration
- Enables large-scale use of design patterns
  - Provides a path for legacy system integration through wrapping

Implemented CORBA Services

- Naming - directory service: (svc name) ➔ (svc object reference)
- LifeCyle - server object management (Factory pattern)
- Event - Producers notify consumers using events
- Trading - service discovery: (svc attributes) ➔ (svc name)
- Transactions - distributed (2PC), flat transactional objects
- Security - specs only. IIOP over SSL currently used

Combinations of these services are implemented in:

- Orbix / IONA (http://www.iona.com)
- Visibroker / INPRISE (http://www.inprise.com/visibroker)
- M3 / BEA (http://www.beasys.com/)
- TAO / UWSTL) (http://www.wustl.edu/~schmidt/ACE.html)

Interoperable Object References (IORs)

- Unique identity of and object, used by clients to invoke its operations
  - can be passed to clients:
    - as parameters or results of operations
    - as strings (see IORs)
- IORs contain:
  - name and location of the object implementation
  - interface type of the object
  - unique key (within the scope of a server)

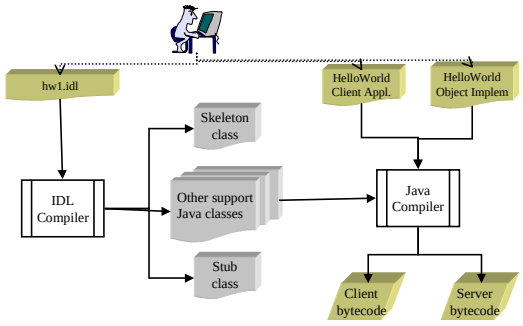
CORBA Object Model Concepts

- Object Implementations vs. Object References
- Types:
  - all types are derived from the root type *Object*.
  - structured types and template types are available
- Interfaces: objects have single interfaces
- Operation Semantics:
  - at-most-once
  - best-effort
  - Operation Signatures may include additional clauses: *raises*, *oneway*, *context*
- Attributes
  - attribute X is logically equivalent to a *getX()*, *setX()* operation pair
- Exceptions: system-defined and user-defined (extensible)

A simple IDL for the lab: HelloWorld1

```
// the simplest IDL example
  interface HelloWorld1 {
     // one method, no parms
     string  hello();
  };
```

Lab: Hello World1: files



Lab: Hello World1: writing the client

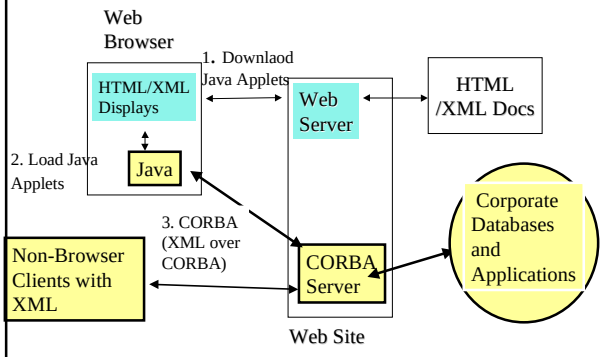
- We need:
  - client-side classes from the IDL compiler

```
 HelloWorld1 hw1 = null;  // interface object
// init ORB
ORB orb = ORB.init(args, null);  // args from command line

// obtain an object reference by binding to the server
// use logical server name and specify target host
hw1 = HelloWorld1Helper.bind("HelloWorld1",host);

// use the reference
if (hw1 != null)
        System.out.println("Server said: "+hw1.hello());
```


Second generation Architectures (Combing Web, CORBA and XML)



CORBA Evolution and Summary

- **CORBA Services**
  - Naming - directory service: (svc name) ➔ (svc object reference)
  - Trading - service discovery: (svc attributes) ➔ (svc name)
  - Event/notifcation - Producers notify consumers using events
  - Transactions - distributed (2PC), flat transactional objects
  - Security - different levels (IIOP over SSL, additionally used)
  - Messaging - Support asynchronous processing for loosely coupled
- **Distribution protocol**
  - GIOP (Generalized Interorb Protocol) and its mappings (IIOP)
  - Mappings for SS7, ATM
- **Specialized Models**
  - Real-time, Fault-tolerant, Minimum CORBA
- **Vertical Domain facilities (e.g., Telecom)**
  - CORBA to TMN Interworking
  - Wireless CORBA
- **Support for Analysis and Design**
  - UML(universal Modeling Language)
- **Basic Object-Oriented Computing Model**
  - CORBA Components, ISO/OMG IDL and language mappings

CORBA Versus DCOM

Why Worry?

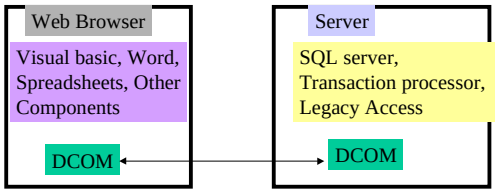
- CORBA and DCOM are two key alternatives for distributed object computing
- CORBA is based on open standards
- DCOM is supported by Microsoft

Analysis should involve:

- Technical assessment
- Industry assessment
- At present both are included in newer platforms (e.g., DCOM in .NET, COBA in J2EE)

DCOM (Distributed Component Object Model)

- Specified by Microsoft.
- Key Component of Microsoft Strategy



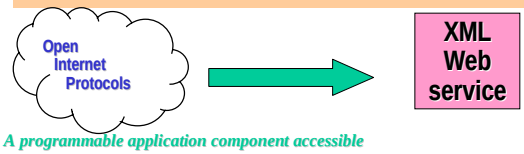
High Level Comparison of CORBA and DCOM

	DCOM	CORBA
Backer	Microsoft	700+ Member Consortium
Services	Well-defined semantics	More
Openness	Proprietary, but becoming less so	Open
Portability	Mostly NT 4.0	20+ platforms
Ease-of-use	Complex, but good tools	Simpler

CORBA VERSUS DCOM SUMMARY

- Similarities:
  - Both are based on the object model
  - Both utilize the interface concept and utilize an Interface Definition Language (IDL).
  - Both use static and dynamic calls from clients to servers
  - Both use a repository to locate objects and invoke them
- Dissimilarities:
  - CORBA is a specification but DCOM is an implementation
  - DCOM uses the universal unique ID (UUID), based on OSF DCE, to locate and invoke objects. CORBA does not use UUIDs.
  - DCOM uses the OSF DCE RPC but CORBA uses mainly IIOP (Internet Inter-ORB Protocol)

XML-Web Services and XML/SOAP



A programmable application component accessible via standard Web protocols

- **Provide a directory of services on the Internet**
- **XML Web services are defined in terms of the formats and ordering of messages**
- **XML Web service consumers can send and receive messages using XML**
- **Built using open Internet protocols**

UDDI
Universal Description, Discovery and Integration

WSDL
XML Web services Description Language

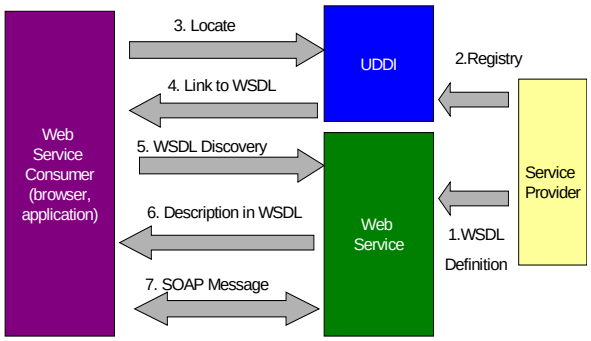
SOAP

XML & HTTP

Standards Used in Web-XML Services

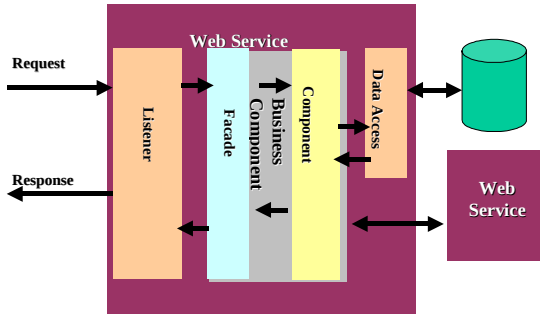
- SOAP (Simple Object Access Protocol)
  - Explicit serialization (HTTP + XML description) protocol used in service exchanges
- WSDL (Web Service Description Language)
  - XML document describing the location and interfaces a particular service supports – the client's contract
- DISCO (Discovery)
  - XML document describing (URI) of service
- UDDI (Universal Description Discovery and Integration)
  - Yellow pages directory for services

Web Services (In Practice)

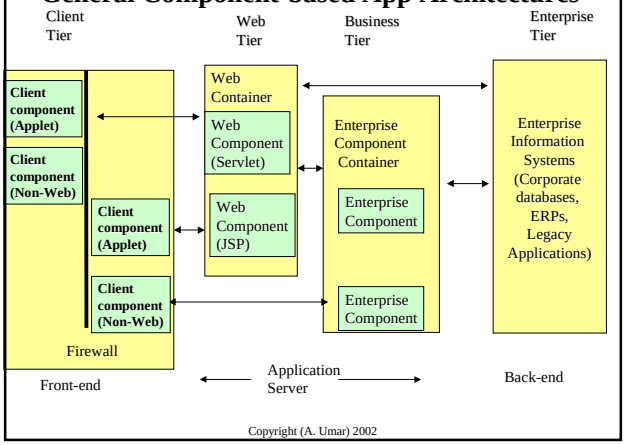


Overview of Web Services

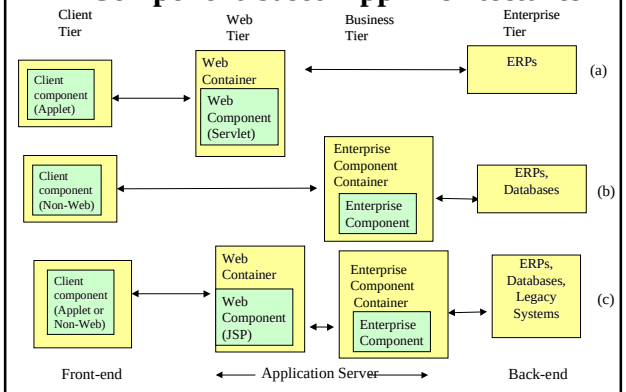
Web Service Architecture



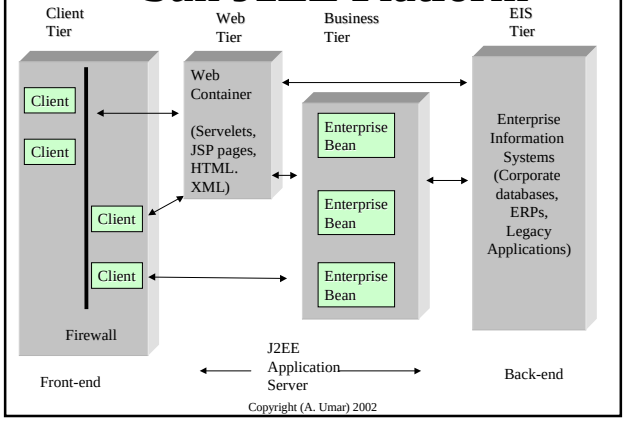
General Component-based App Architectures



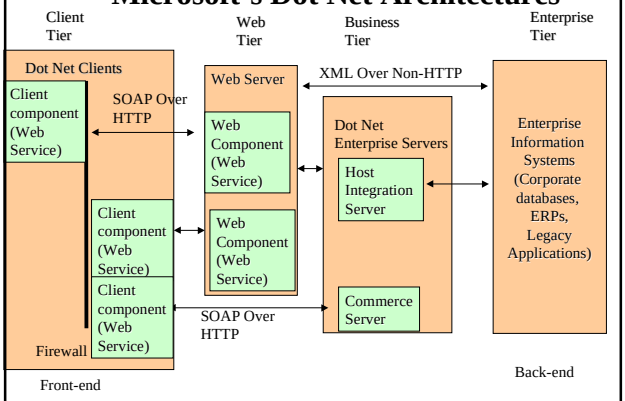
Component-based App Architectures



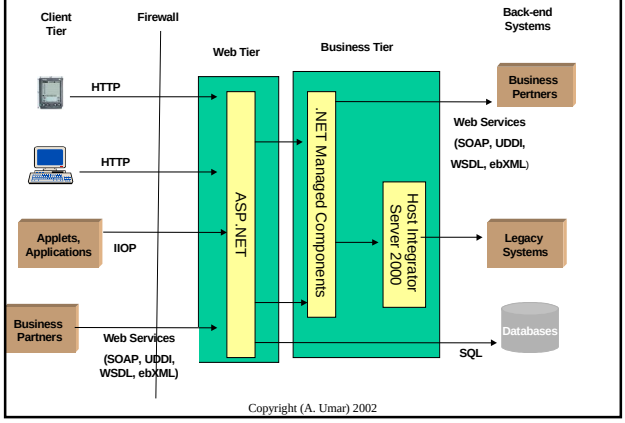
Sun J2EE Platform



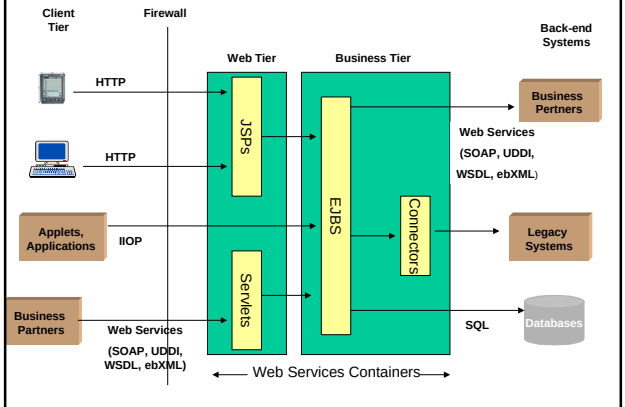
Microsoft's Dot Net Architectures



Web Services in .NET



Web Services in J2EE



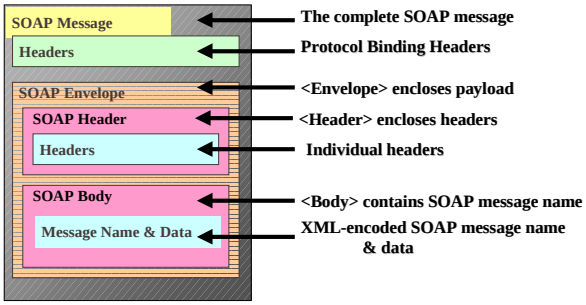
SOAP for Distributed Objects

SOAP = HTTP + XML

- SOAP is part of Web Services and thus belongs to the .NET as well as J2EE-based platforms
- A Lightweight Protocol For Exchanging Information In A Distributed, Heterogeneous Environment
- The SOAP Specification Defines
  - The SOAP message format
  - How to send messages
  - How to receive responses
  - Data Encoding

SOAP

Message Structure



SOAP

Security and Features

- Builds on HTTP Security
  - HTTPS
  - X.509 certificates
- Developers / IT Choose Which Methods to Expose Explicitly
- Does Not Pass Application Code
- Firewall-Friendly
- Type Safe