Middleware
Lou Somers

April 18, 2002

Contents

- Overview
  - Definition, goals, requirements
- Four categories of middleware
  - Transactional, message oriented, procedural, object
- Middleware examples
  - XML-RPC, SOAP, Corba, (D)COM
- Some applications
  - Controller - engine communication
  - Controller architecture
  - Web based printer monitoring and control
History

- **Mainframes**
  - Centralized computing
  - Expensive devices
- **Client-server paradigm**
  - More functions performed by client PC's
  - Applications split into light client part and sharable server part
- **Distributed objects**
  - Distribution combined with objects
  - Components can have both server and client roles

Why distributed?

- **Increased resource sharing**
- **Increased maintainability and reduced costs**
  - Install once, run anywhere
  - Upgrade software on single machine only
- **Increased fault tolerance**
  - Replication
- **Performance**
  - Compared to single machine (-)
  - Compared to client-server (+)
What is middleware?

- “The glue which connects objects which are distributed across multiple heterogeneous computer systems”
- “An extension of the operating system which provides a transparent communication layer to the applications”
- “A software layer that serves to shield the application of the heterogeneity of the underlying computer platforms and networks”

Middleware goals

- **Integrate existing components into a distributed system**
  - Components may be off-the-shelf
  - Components may have incompatible requirements for hardware and OS platforms
  - Scalability requires distribution (not centralized or client server).
- **Resolve heterogeneity**
  - Facilitate communication and coordination of distributed components
  - Build systems distributed across a local area network, the internet
  - Future: adaptive, reconfigurable
Requirements for middleware / 1

- **Network communication**
  - Need higher level primitives than network operating system primitives
  - Transport complex data structures over the network (marshalling / unmarshalling)

- **Coordination**
  - Three models:
    - Synchronous: client waits for result
    - Deferred synchronous: client asks for result (e.g. by polling)
    - Asynchronous: server initiates result
  - Group requests
  - Component activation / deactivation
  - Concurrent requests

Requirements for middleware / 2

- **Reliability**
  - Error detection and correction mechanisms on top of network protocols

- **Scalability**
  - Access to a component independent of whether it is local or remote
  - Migration transparency
  - Replication transparency

- **Heterogeneity**
  - Primitive data encoding
  - Different programming languages
### Middleware categories

- **Transactional middleware**
  - Offers a *tuple* abstraction (SQL)
  - Distributed transaction processing (DTP protocol)

- **Message oriented (MOM)**
  - Offers a *mailbox* abstraction
  - Asynchronous messages

- **Procedural (RPC)**
  - Offers a *procedure* abstraction
  - Synchronous client / server interaction

- **Object and component**
  - Offers an *object* abstraction
  - (A)synchronous client / server interaction

### Transactional middleware / 1

- **Transactions on distributed relational database**
  - Two-phase commit protocol to implement distributed transactions

- **Examples**
  - IBM CICS
  - BEA Tuxedo
Transactional middleware / 2

- **Network communication**
  - Client and servers may reside on different hosts

- **Coordination**
  - Synchronous and asynchronous

- **Reliability**
  - DTP (Distributed Transaction Protocol): two phase commit
  - ACID properties:
    - Atomic: transaction is either complete or not
    - Consistent: system always in consistent state
    - Isolation: transaction is independent of other transactions
    - Durable: committed transaction survives system failures

- **Scalability**
  - Load balancing and replication of server components

- **Heterogeneity**
  - Different hardware and operating systems platforms
  - No data heterogeneity

Message oriented middleware / 1

- **Exchange messages between components**
  - “Mailbox”

- **Examples**
  - Java Message Queue
  - IBM MQSeries
Message oriented middleware / 2

- **Network communication**
  - Client sends message, server replies with result
  - Well suited for event notification and publish / subscribe

- **Coordination**
  - Asynchronous
  - Synchronous has to be coded by client

- **Reliability**
  - Message queues are stored on persistent memory
  - At-least-once semantics possible

- **Scalability**
  - Local / remote differs

- **Heterogeneity**
  - Marshalling code has to be written by hand

Procedural middleware / 1

- **Remote procedure calls (RPC)**
  - Procedures can be called across the network

- **Examples**
  - Unix RPC’s
  - DCE RPC (Distributed Computing Environment)
  - Windows RPC’s
  - XML-RPC
**Procedural middleware / 2**

- **Network communication**
  - Server exports parameterized procedures
  - Clients call these across the network
  - Marshalling and unmarshalling by client and server stubs (generated by the compiler)

- **Coordination**
  - Synchronous interaction between one client and one server
  - Startup on demand possible (daemon needs a table that maps RPC names to program locations in the file system)

- **Reliability**
  - At-most-once semantics (exception if RPC fails)

- **No scalability**

- **Heterogeneity**
  - Can be used between different programming languages
  - Across different hardware and OS platforms

---

**Object middleware / 1**

- **Objects are available across the network**

- **Examples**
  - Corba
  - COM, DCOM
  - Java RMI, Enterprise Java Beans
Object middleware / 2

- **Network communication**
  - Client objects call methods of exported server objects
  - Marshalling and unmarshalling by client and server stubs
    (generated by the compiler)

- **Coordination**
  - Default: synchronous
  - Corba 3.0: also deferred synchronous and asynchronous

- **Reliability**
  - Default: at-most-once semantics (exceptions on failure)
  - Usually requests may be clustered into transactions
    (Object Transaction Server, Microsoft Transaction Server, Java Transaction Service)

- **Scalability**
  - Support for load-balancing, replication is rather limited

- **Heterogeneity**
  - Corba, COM: multi-language binding
  - Corba / RMI and Corba / COM may interoperate

Middleware examples

- **XML-RPC, SOAP**
  - RPC on top of HTTP

- **Corba**
  - Distributed objects

- **COM / DCOM**
  - Microsoft’s Component Object Model
XML-RPC protocol / 1

- Uses XML to transmit and receive RPC’s
  - Defines the bare minimum to get RPC’s across the network
- Based on HTTP with the POST method
  - Request is an XML document containing a method name and parameters
  - Response is an XML document with returned values

- SOAP (Simple Object Access protocol)
  - Successor of XML-RPC

XML-RPC protocol / 2

- HTTP Header
  - Content-Type: text/xml
  - Content-Length must be specified and correct
  - User agent must be specified
  - Host must be specified
- Document content
  - Root <methodCall>
  - Contains name of method: <methodName>
  - Contains list of parameters: <params>
  - Each parameters is a pair <param>, <value>
- Data types supported
  - String, integer, float, date, binary, boolean
  - Array, struct
Corba

- “Common Object Request Broker Architecture”
- Defined by Object Management Group (OMG)
  - > 800 companies: www.omg.org
- Goal: enable interoperability
  - Languages, implementations, platforms
- Conceptual view:

Corba: overall architecture

- Static interfaces
  - Specific for each object type
- Dynamic interfaces
  - Same interface independent of target object
- Object adapter
  - Object references, object (de)activation, method invocation, ...
Corba: general interaction scheme

- **Stubs and skeletons**
  - Generated from the Interface Definition (in IDL)
  - Using appropriate language mappings

- **Interface Definition Language**
  - Purely descriptive, not an implementation language
  - Strongly typed

Corba: object model

- **Object**
  - Identifiable entity
  - Provides services to a client
  - May be created and destroyed
  - The interface repository contains type information

- **Service requests**
  - Consist typically of: operation, target object, arguments
  - May be specified statically or formed dynamically

- **Values**
  - Non-object values and object references

- **Inheritance**
  - An interface may inherit from another interface
  - Multiple inheritance
  - **Object**: base of the entire hierarchy
Corba: object request broker

- **More realistic view:**
  - Diagram of client, server, and ORB connections.

- **Protocol between ORB’s**
  - General InterORB Protocol (GIOP)

- **GIOP on top of TCP/IP**
  - Internet InterORB Protocol (IIOP)

- **Future: on top of SOAP**

---

**COM / DCOM**

- **(Distributed) Component Object Model**
  - Functionality like Corba
  - Internally, (D)COM uses RPC’s

- **Difference between interface and implementation**
  - Objects may implement more than one interface
  - Interface is immutable
  - Objects may be implemented in “any” language

- **Objects register themselves in the Windows registry**
  - Using a 128-bit Global “Unique” Identifier (GUID)

- **IDL compiler produces a type library**
  - Binary description
  - Usable from all COM aware languages
  - Registered into the registry
Some middleware applications

- **Controller - engine communication**
  - Via message oriented middleware
- **Controller architecture**
  - Components interact via COM
- **Web based printer monitoring and control**
  - ASP’s use COM

Controller - engine communication

- **Control via “CSL” middleware (Client Server Layer)**
  - Message oriented
  - Synchronous (post message)
  - Asynchronous (send message, subscribe)
Controller architecture

- Components interact via COM with blackboard

Web based printer monitoring / control

- Active Server Pages interact via COM with “business objects”
- RPC and SNMP used for interaction with printers