object space for sharing and exchanging objects between components of a distributed application

JavaSpaces supports an object space.

based on the Linda tuple concept.

Tuples are references to Java objects

Example Java Spaces
Object Space

Object-based Distributed Systems

Introduction
Features of JavaSpaces
Data structures
Entry interface
SpaceAccesser
Basic operations
Events
Example Java Spaces

Object Management Architecture - OMA

The Object Management Group (OMG) was founded in 1989 by a number of companies to encourage the adoption of distributed object systems and to enable interoperability for heterogeneous environments (hardware, networks, operating systems and programming languages).

Object Management Architecture - OMA
Object Request Brokers ORB
Common object services
Inter-ORB protocol
Distributed COM
.NET Framework

The OMG (Object Management Group) was founded in 1989 by a number of companies to encourage the adoption of distributed object systems and to enable interoperability for heterogeneous environments (hardware, networks, operating systems and programming languages).

Object Management Architecture - OMA
Object Request Brokers ORB
Common object services
Inter-ORB protocol
Distributed COM
.NET Framework
General features

ORB supports the following general characteristics:

1. static and dynamic invocation of object methods
   - **static**: method interface is determined at compile time.
   - **dynamic**: method interface is determined at runtime.
2. interfaces for higher programming languages, e.g., C++, Smalltalk, Java.
3. a self-descriptive system.
4. location transparency.
5. security checks, e.g., object authentication.
6. polymorphic method invocation, i.e., the execution of the method depends on the specific object instance.
7. hierarchical object naming.

Structure of ORB

- ORB core (kernel): mediates requests between client and server objects; handles network communication within the distributed system.
- ORB interface: supports ORB service calls, e.g., conversion of object references to strings and vice versa; the interface is determined by the ORB.
- Interface repository: stores at runtime the signatures of the available methods; the signatures are described by the IDL notation; in case of the dynamic invocation interface a lookup within the interface repository is performed.
- Object Adapter: bridges the gap between Corba objects with IDL interfaces and the programming language interfaces of the server class.

Embedding in distributed Applications

- ORB core (kernel): mediates requests between client and server objects; handles network communication within the distributed system.
- ORB interface: supports ORB service calls, e.g., conversion of object references to strings and vice versa; the interface is determined by the ORB.
- Interface repository: stores at runtime the signatures of the available methods; the signatures are described by the IDL notation; in case of the dynamic invocation interface a lookup within the interface repository is performed.
- Object Adapter: bridges the gap between Corba objects with IDL interfaces and the programming language interfaces of the server class.
Embedding in distributed Applications

Usually the ORB is embedded as a library function.

- ORBIX by Microfocus
- TAO system by Doug Schmidt

Inter-ORB protocol

Communication between ORBs is based on GIOP ("General Inter-ORB Protocol").

GIOP Features
- External data representation
- Object reference
- GIOP message
- Example for IIOP use
- RMI over IIOP

Common object services

A collection of system level services which can be utilized by the application objects; they are extending the ORB functionality.

- Life-cycle Service: defines operations for object creation, copying, migration and deletion.
- Persistence Service: provides an interface for persistent object storage, e.g. in relational or object-oriented databases.
- Name Service: allows objects on the object bus to locate other objects by name; integrates existing network directory services, e.g. OSI's DCE, LDAP or X.500.
- Event Service: registers the interest in specific events; producer and consumer of events need not know each other.
- Concurrency Control Service: provides a lock manager.
- Transaction Service: supports 2-phase commit coordination for flat and nested transactions.
- Relationship Service: supports the dynamic creation of relations between objects that know nothing of each other; the service supports navigation along these links, as well as mechanisms for enforcing referential integrity constraints.
- Query Service: supports SQL operations for objects.

External data representation

Distinction between primitive and complex data types, so-called typeCodes; assignment of integer values to identify data types.

- primitive: char, octet, short, long, float, double, boolean
- complex: struct, union, sequence, symbol chains, fields

The format of complex data types is described in the interface repository.

Example

tk_struct (Typecode struct):
string : repository_ID
string : name
ulong: count

  {string : member name
   TypeCode: membertype }
Communication between ORBs is based on GIOP ("General Inter-ORB Protocol").

**GIOP Features**
- External data representation
- Object reference
- GIOP message
- Example for IIOP use
- RMI over IIOP

---

**GIOP message types**

GIOP supports the following message types:

1. **Request**: request the execution of an operation at the remote object, e.g., access of an attribute; the message contains the call parameters.
2. **Reply**: answer to a request message.
3. **CancelRequest**: termination of a request; the calling ORB does not expect an answer to the original request.
4. **LocateRequest**: is used to determine whether the given object reference is valid, or whether the destination ORB processes the object reference, and if not, to which address requests for the object reference are to be sent.
5. **LocateReply**: answer to LocateRequest
6. **CloseConnection**: the destination ORB notifies the calling ORB that it closes the connection.
7. **MessageError**: exchange of error information.
8. **Fragment**: if, for instance, the request consists of several parts, then first a request message is sent, and then the remaining parts are sent using fragment messages.

---

**GIOP message head**

The GIOP message head has the same format for all message types; it identifies the message type sent to another ORB.

**GIOP message head structure**

```c
module GIOP

struct Version (octet: major; octet: minor);

enum MsgType (Request, Reply, CancelRequest, LocateRequest, LocateReply, CloseConnection, MessageError, Fragment)

struct Message_Header {
    char magic[4]; this is the string "GIOP"
    Version GIOP_version;
    octet flag
    octet message_type
    unsigned long message_size
}
```

The component `flag` determines the used byte ordering (big/little endian) and whether or not the entire message has been divided into several fragments. `message_type` is an element of the `MsgType` enumeration; it identifies the message type.

**Example for IIOP use**

Web access of a database using a Java applet and Corba.
RMI over IIOP

RMI uses JRMP (Java Remote Method Protocol) for the communication between client and server objects, i.e. there is no interoperability with Corba.

Corba client (Java) \(\xrightarrow{\text{JRMP}}\) Corba server (Java)

Corba client (any programming language) \(\xrightarrow{\text{IIOP}}\) Corba server (any programming language)

Extension of RMI to RMI-IIOP

---

Distributed COM

DCOM grew out of COM (Component Object Model) tightly integrated into Windows OS.

- **Goal** of COM: support the development of components that can be dynamically activated and that can interact with each other.
- **component**: executable code either contained in a DLL or in form of an executable program.
- **COM**: offered in form of a library that is linked to a process.
- **DCOM**: extension of COM which allows a process to communicate with components that are placed on another machine.
  - DCOM provides access transparency.

Object Model

Architecture

Object Invocation Model

DCOM was combined with Microsoft Transaction Server (MTS) and Microsoft Message Queue Server (MSMQ) to COM+

- Supports distributed transactions to enable transactional distributed applications integrated into Windows OS
The overall architecture of DCOM in combination with the use of class objects, objects and proxies has the following form:

The type library specifies the exact signature of the method to be invoked dynamically, the registry records the mappings of a call identifier to a local file name containing the implementation of that class, the service control manager (SCM) is responsible for activating objects, and the proxy marshaler deals with transforming the code of a proxy into a series of bytes for network transmission.

Distributed COM grew out of COM (Component Object Model) tightly integrated into Windows OS.

**Goal** of COM: support the development of components that can be dynamically activated and that can interact with each other.

- **Component**: executable code either contained in a DLL or in form of an executable program.
- **COM**: is offered in form of a library that is linked to a process.
- **DCOM**: extension of COM which allows a process to communicate with components that are placed on another machine.

DCOM provides access transparency.

Object-based Distributed Systems

The OMG (Object Management Group) was founded in 1989 by a number of companies to encourage the adoption of distributed object systems and to enable interoperability for heterogeneous environments (hardware, networks, operating systems and programming languages).
Common Language Runtime (CLR)

- Provides a runtime environment for applications which may be developed in different languages, e.g., C#, C++, Java, Perl or Python. CLR supports the following services:
  - Memory management.
  - Thread management.
  - Libraries that encapsulate access to OS functions.
  - Common Intermediate Language (MSIL).

All .NET programs execute under the supervision of the CLR.

Common Type System (CTS)

- CTS defines all possible datatypes and programming constructs supported by the CLR.
- Data structures are uniformly interpreted on the MSIL layer.
- Enables interoperability between the languages supported by .NET.

Frame Class Library

- Object-oriented library of common functions available to all languages using the .NET Framework.
- Enables file access, XML document manipulation, or database interaction.
- Organized in a hierarchy of namespaces.

System::Object is the base of all library classes and application classes.

.NET Framework

- The Microsoft .NET Framework is a software framework available with Windows OS for building distributed applications.
- Represents a strategy change from the product-oriented desktop world to the service-oriented component world.
- Goal: Many future Windows applications should be built using .NET.

.NET has 2 core elements: Common Language Runtime (CLR) and the Framework Class Library.

Common Language Runtime (CLR)

.NET Remoting

- Technology for remote method invocation provided by the framework.
- Relevant classes are in the namespace `System.Runtime.Remoting`.
- Support of different transport protocols, e.g., TCP (binary formatting) or HTTP (SOAP formatting).
- Activation of remote objects.

Distributed Applications - Verteilte Anwendungen

- Prof. J. Schlichter
  - Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Fakultät für Informatik, TU München
  - Boltzmannstr. 3, 85748 Garching
  - Email: schlchter@in.tum.de
  - Tel.: 089-289 18654
  - URL: http://www1.in.tum.de/

Overview

- Introduction
- Architecture of distributed systems
- Remote Invocation (RPC/RMI)
- Basic mechanisms for distributed applications
- Web Services
- Design of distributed applications
- Distributed file service
- Distributed Shared Memory
- Object-based Distributed Systems
- Summary