E Distributed Objects in CORBA

E.1 Overview

- Motivation
- Architectural Overview
- Application Objects
- Object Request Broker
- Portable Object Adaptor
- CORBA Services

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.2 References

OMG99. Object Management Group, OMG: *The Common Object Request Broker: Architecture and Specification.* Rev. 2.3.1, OMG Doc. formal/99-10-07, Oct. 1999.

Pope98. A. Pope: *The CORBA Reference Guide.* Addison-Wesley. 1998.

Linn98. C. Linnhoff-Popien: *CORBA, Kommunikation und Management.* Springer, 1998.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

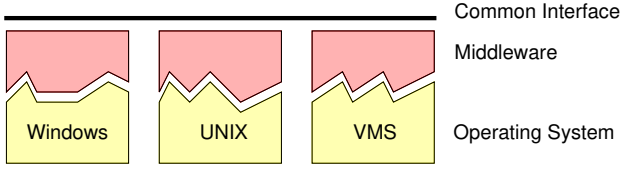
E.3 Motivation

- Location
  - Transparency of location
- Heterogeneity
  - Different hardware,
  - different operating systems, and
  - different programming languages are used in distributed systems.
  - Transparency of heterogeneity
- Services
  - Name server
  - Time server
  - ...

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Middleware Approach

- Middleware: a piece of software between operating system and application



  - Middleware provides services for distributed programming
- **We need:** middleware for distributed object-based programming
- ★ CORBA – Common Object Request Broker Architecture, a standard of the OMG

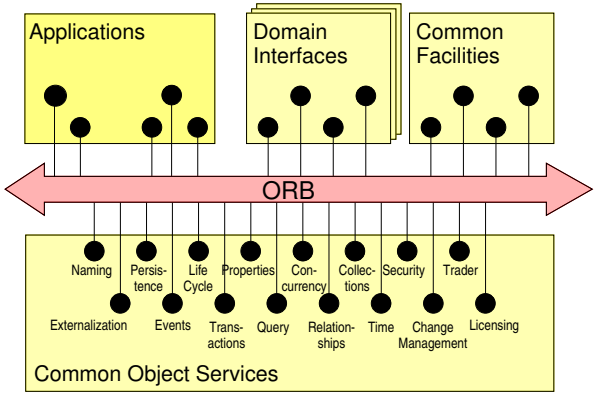
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 CORBA Design Goals

- CORBA is a standard
  - There are standard documents
  - Definition of CORBA compliance
  - Vendors build implementations of the standard (e.g., VisiBroker, Orbix, Orbacus, MICO, etc.)
- CORBA builds abstractions from
  - hardware,
  - operating system, and
  - programming language.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.4 Architectural Overview



- Object Management Architecture – OMA

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 CORBA Design Goals (2)

- Interoperability
  - An application shall run on different CORBA implementations without major changes
  - Applications on different CORBA implementations shall be able to communicate
- Embedding of legacy applications
  - Legacy applications can be encapsulated and can act as CORBA objects
- Vision of business objects or components
  - CORBA objects represent all kinds of business data in a company
  - By communicating to these objects business data can be manipulated

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 OMA – Object Management Architecture

- Application objects
  - Centralized-object approach for distributed objects
  - Client/server architecture
- Object Request Broker (ORB)
  - Communication backbone
  - Enables objects to communicate
- CORBA Services
  - Basic services for distributed programming
  - Extends the ORB's functionality
  - Services look like objects

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 OMA – Object Management Architecture (2)

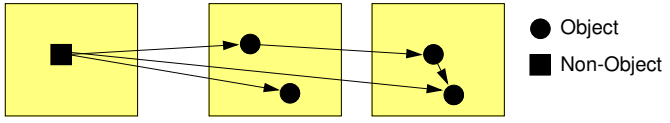
- CORBA Facilities
  - Application-specific services
  - Usable in multiple application domains
  - e.g., Compound Document Services, Graphical User Interface
- CORBA Domains
  - Application-specific services for a particular domain
  - e.g., CORBAmed

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.5 Application Objects

1 Distributed Object

- Identity
- State
- Operations (methods)
- CORBA objects can be invoked (implement server side)
- CORBA objects may act as clients



- Clients do not need to be CORBA objects
  - CORBA clients can be processes, etc.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 CORBA Implementations

- A CORBA implementation must contain
  - the implementation of the core architecture
  - one language mapping (e.g., for C++)
- A CORBA implementation may contain
  - an arbitrary number of services
  - functionality for interoperability with other CORBA implementations (GIOP/IIOP)
- The implementation is free **how** it realizes the requirements of the standard
  - Many different implementations are possible
    - Daemon-based
    - Library-based
    - ...

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

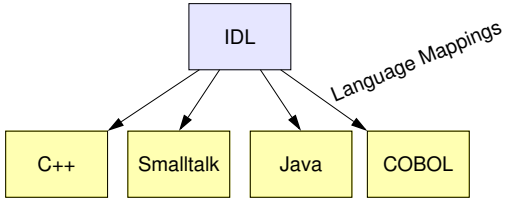
2 Distributed Object (2)

- Distributed objects form an application
  - Objects cooperate by communicating with each other
- ★ Example: printer management system
  - Client objects
  - Spooler objects
  - Printer objects
- "Centralized object approach"
  - Client-side stub can contact the server object
  - RPC-based method invocation
  - **At-most-once** / **exactly-once** semantics

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interface Definition Language (IDL)

- Language for describing object interfaces
  - Independent of implementation language of an object
  - Language mapping defines how IDL primitives are mapped to the concepts of a specific programming language
  - Language mapping is part of the CORBA standard
  - Language mappings exist for C, C++, Smalltalk, COBOL, Ada, and Java
  - IDL is similar to C++



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interface Definition Language (2)

- Example

```
module MyModule
{
    interface MyInterface
    {
        attribute long lines;
        void printLine( in string toPrint );
    };
```

IDL to C++

```
namespace MyModule {
    class MyInterface : ... {
        public:
         virtual CORBA::Long lines();
         virtual void lines( CORBA::Long _val );
         void printLine( const char *toPrint );
        ...
    };
```


Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interface Definition Language (3)

- Example

```
module MyModule
{
    interface MyInterface
    {
        attribute long lines;
        void printLine( in string toPrint );
    };
```

IDL to Java

```
package MyModule;
public interface MyInterface extends ... {
    public int lines();
    public void lines( int lines );
    public void printLine(java.lang.String toPrint );
    ...
};
```


Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interface Definition Language (4)

★ Advantages
- Transparency of implementation language
- Enables interoperability

▲ Disadvantages
- Object interface must be defined in the target language **and** in IDL
- IDL is quite expressive
  - Language that do not provide the right mechanisms will have a complex language mapping (e.g., sequences)
- If a language has special features they cannot be used because they are not part of IDL

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

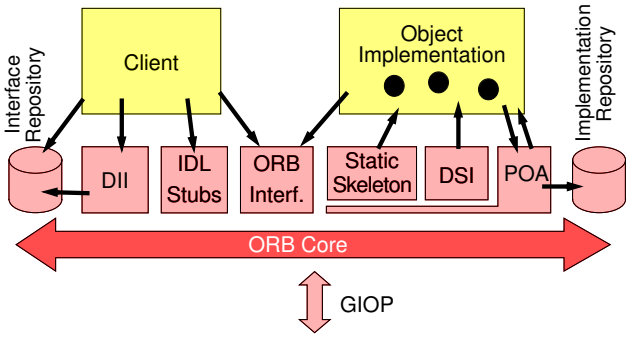
4 Process of Creation and Binding

- Creation of a server object
  - Description of the object interface in IDL
  - Programming the server object in an implementation language
  - Registration of the object at the ORB (the OA respectively)
    - The ORB creates an Interoperable Object Reference (IOR)
- Binding to the server object at client side
  - Retrieval of an object reference to the server
    - Result of a name-server query
    - Return parameter of a method call
    - Retrieval from outside of the system: user knows the reference as a string (IORs can be converted to strings and back)
  - Creation of the client stub
  - Method invocations using the stub object

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

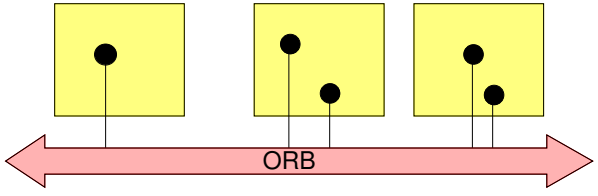
1 Architecture

- Central components of an ORB



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.6 Object Request Broker – ORB



- The ORB is the communication backbone of a CORBA implementation
- All communication is handled by the ORB
  - ... within an address space / process
  - ... between address spaces / processes
  - ... between address spaces / processes of different ORBs
- The ORB implements location transparency

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

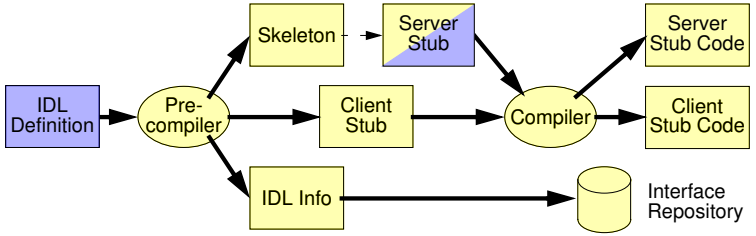
2 Static Stubs

- Stubs on client and server side
  - As soon as we exactly know the interface of an object, we can create static stubs from it.
  - Static stubs are automatically created from the IDL description
  - Server side stubs are called *skeletons*
    - CORBA skeletons have to be filled with the actual implementation
- Tasks of the stubs
  - Marshalling of parameters
  - Send and receive of request and reply messages using the ORB core

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Static Stubs (2)

- Stub creation



  - From IDL, a precompiler generates a number of output files.
  - The skeleton contains a frame for filling in the actual methods of the object implementation. The skeleton already contains the server-stub functionality.
  - The client-stub file contains the code for the client stub.
  - An additional file may contain data for the interface repository.
  - Usually the stubs have to be compiled by the target-language compiler.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

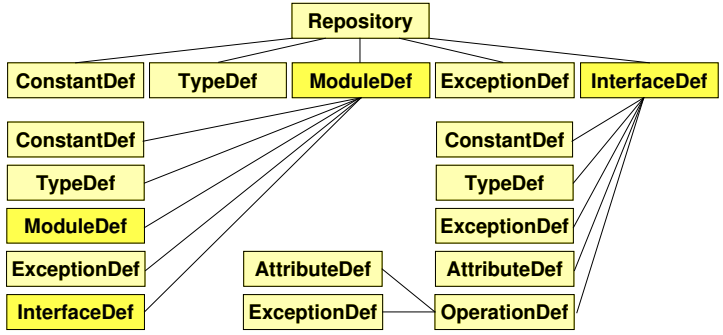
3 Interface Repository

- Database for interface definitions in IDL
- Query the database for
  - Type checking
    - Does the stub's type match the object's type?
    - Inter-ORB operations: insertion into multiple repositories
  - Retrieval of meta data by clients and tools
    - Dynamic invocations
    - Debugging
    - Class browser
  - Implementing method *get_interface* of each object
- Writing to the database using
  - IDL compiler
  - Write methods

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interface Repository (2)

- Stored information / composition of an IDL file

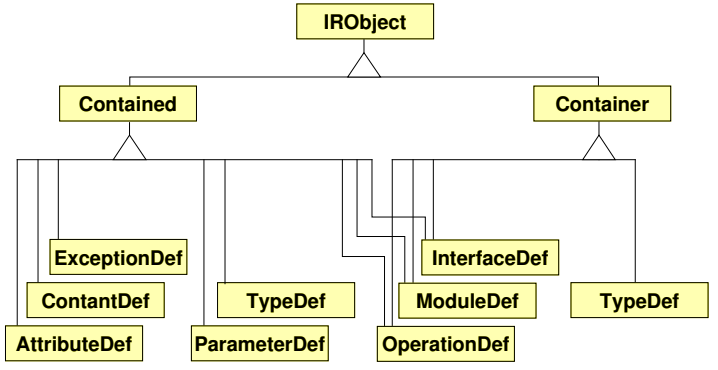


  - Hierarchy of the components of an IDL file, and of the interface repository respectively

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interface Repository (3)

- Inheritance hierarchy of the IDL interfaces of IR objects



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Interface Repository (4)

- Standard types are represented by so-called *type codes*

TypeCode

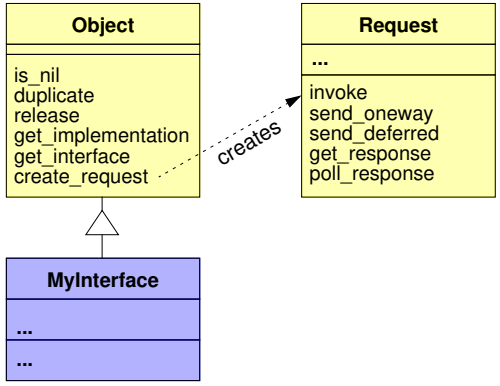
  - Representation of basic types: **int, float, boolean**
  - Representation of composite standard types: **union, struct, enum**
  - Representation of template types and complex declarators: **sequence, string, array**
  - Representation of IDL-based object types (using an interface repository ID)

- Type codes represent type and structure as objects with an standard IDL interface
  - Operation for comparing type codes
  - Operation for obtaining the description of the type

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Dynamic Invocation Interface (2)

- IDL definitions for the DII



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Dynamic Invocation Interface (DII)

- DII allows the invocation of methods whose object interface was not known at compile time
  - Interface description can be retrieved using the interface repository
- Single steps of an invocation (which have to be mapped by the language binding)
  - Retrieve signature of the method from the interface repository
  - Create list of parameters
  - Compose request
  - Invoke method
    - with RPC (synchronous call)
    - with asynchronous RPC (asynchronous call)
    - with datagramm message (no reply, no reliability)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Dynamic Skeleton Interface (DSI)

- DSI allows a CORBA server to accept method invocations for objects whose interfaces are only dynamically retrievable, e.g., in
  - Bridges to other ORBs
  - CORBA-encapsulated data bases
  - Dynamically created objects and interfaces
- The DSI identifies the object for which the invocation is made for
  - A call-back function is invoked that gets all the necessary information
  - The call-back function has to invoke the corresponding object

★ **Note:**
DII and DSI are not distinguishable from static stubs, i.e., they are fully interoperable with static stubs.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Object Adaptor

- The object adaptor is a local representative of ORB services
  - It generates object references (for new objects)
  - It maps object references to implementations
  - It forwards incoming method calls
  - It authenticates the caller (implementing CORBA's security functionality)
  - It activates and deactivates the object and its implementation
  - It registers server classes within the implementation repository
- CORBA defines the mandatory Portable Object Adaptor
  - Basic functionality
- Another example: an OODB Adaptor
  - Connecting an object-oriented data base to CORBA
  - The OODB Adaptor represents all objects in the data base as CORBA objects and mediates invocations

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Implementation Repository

- Data base for object implementations
  - Data to localize and activate object implementations
  - Information for debugging and administration
  - etc.
- Implementation repository is highly implementation dependent.
- Query the data base
  - Implementation of the method *get_implementation* of each object
- Writing the data base by
  - External tools
  - The object adaptor at creation time of an object

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

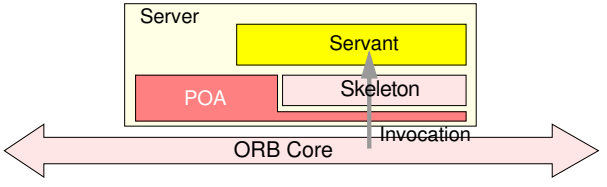
8 Inter-ORB Communication

- GIOP – General Inter-ORB Protocol
  - Basic protocol for the interaction of two ORBs
  - Common Data Representation (CDR) converts IDL-compliant parameters into a serial byte stream
  - IIOP – Internet Inter-ORB Protocol
    (GIOP over TCP/IP; implementation is mandatory)
  - Other implementations of GIOP are possible
- ESIOP – Environment Specific Inter-ORB Protocols, e.g., DCE/ESIOP
  - Inter-ORB protocol on the basis of DCE RPC
- IOR – Interoperable Object Reference
  - String representation
  - Several profiles (one of it is IIOP)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.7 Portable Object Adaptor (POA)

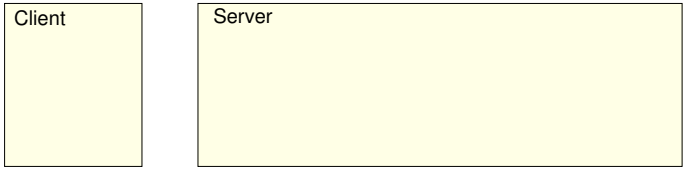
- Terminology
  - **Server:** Process hosting CORBA objects
  - **Servant:** Language object representing the implementation of a CORBA objec
- Each servant knows one POA instance within a server
  - The object reference includes the identifier of this POA instance
  - The POA creates and identifies ***its*** CORBA objects
  - The POA forwards requests to its objects



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Creation of an Object

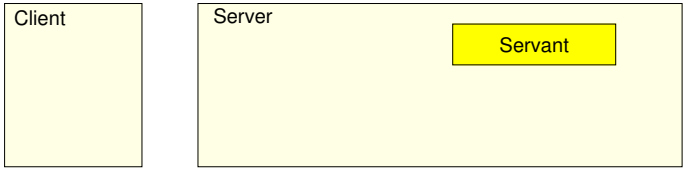
- Application creates an object



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Creation of an Object

- Application creates an object

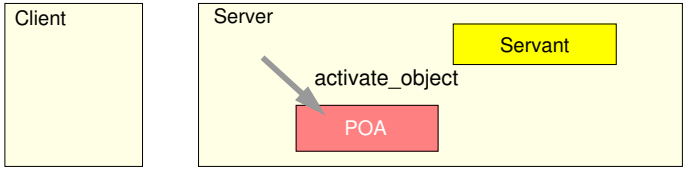


- First, the servant is created

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Creation of an Object

- Application creates an object

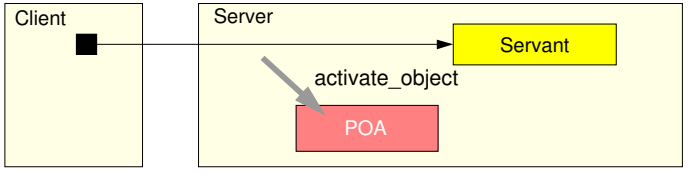


- First, the servant is created
- Second, the CORBA object is activated (can now be remotely invoked)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Creation of an Object

- Application creates an object



- First, the servant is created
- Second, the CORBA object is activated (can now be remotely invoked)
- Then, the servant can be passed as parameter or returned as a result, which creates a client stub (CORBA object reference) at the receiver side

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Alternative Activations

- Implicit activation
  - POA automatically creates an object reference if servant reference is passed as parameter or result
- Activation on demand
  - Object references can be created without having an active servant
  - In case of incoming requests some alternatives are possible:
    - A registered default servant handles the request (e.g., for wrapping database objects)
    - A user-supplied servant manager returns a servant (servant manager can create the servant on demand)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Deactivation and Activation

- Objects may survive a POA instance: persistent objects
  - Servant can be deactivated
  - POA can be deactivated
  - Server can be shut down
- POA cooperates with a location forwarding service and the implementation repository
  - On deactivation the necessary information for activation is retained in the implementation repository
  - Requests for deactivated servants trigger a forwarding service to launch a new server
  - The server gets the forwarded requests and activates a POA for it
  - The POA activates a servant for the request

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Policies

- Policies apply to one POA instance
  - There may be many (named) POA instances within a server each with different policies
- Activation method is encoded in such policies
  - Servant retention policy
  - Request processing policy
  - Implicit activation policy
- Other policies
  - Thread policy: multi-threaded or single-threaded
  - ID uniqueness policy: unique IDs or not
  - ID assignment policy: user- or system-provided IDs
  - Object lifespan policy: transient or persistent

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

E.8 CORBA Services

- Basic services of a distributed system as extensions of the ORB
  - Naming service
  - Transaction service
  - Persistent object service
  - Life cycle service
  - etc.
- Services are accessible by IDL interfaces
  - No additional primitives necessary; method invocation is enough

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

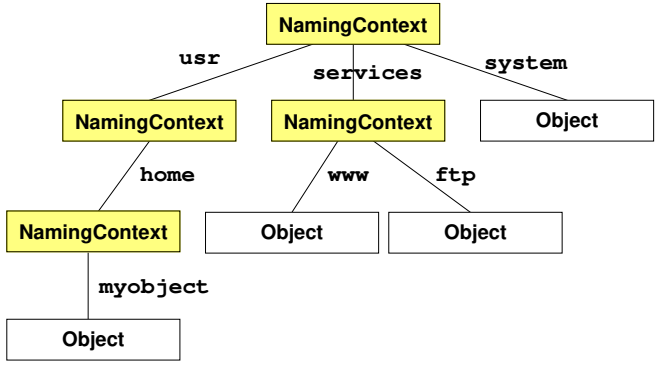
1 Naming Service

- CORBA defines an hierarchical naming service similar to the UNIX file system
  - Name space forms a tree
  - Names consist of multiple components (syllables)
    e.g., < "usr"; "home"; "myobject" >
    (UNIX: "/usr/home/myobject")
  - Interface to the naming service is defined by IDL

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Naming Service (2)

- Example of a name tree



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Naming Service (3)

- Interface of a Naming Context and the Binding Iterator

NamingContext
resolve list destroy new_context unbind bind rebind bind_context rebind_context bind_new_context

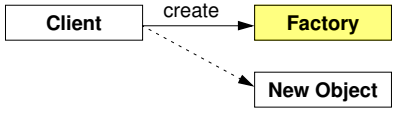
BindingIterator
next_one next_n destroy

  - Naming Context acts like a directory
  - Objects or other contexts can be bound to a name
  - Listing the context returns an array and an iterator that allows you to look at the next entries

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Life Cycle Service

- Controls the life cycle of an object
  - Creation
  - Copying, Migration
  - Deletion
- Life Cycle Service defines the common interface for controllable objects
- ★ Model of the life cycle
  - Creation of an Object:



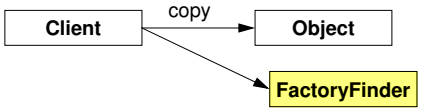
Call of a create method at a factory object

  - Protocol and interface of the factory is not defined and probably dependent on the kind of object to create

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Life Cycle Service (2)

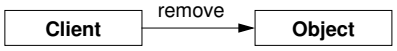
◆ Copying and migration:



FactoryFinder determines the location

• Locations can be a specific computer or a group of computers, etc.

◆ Deletion:

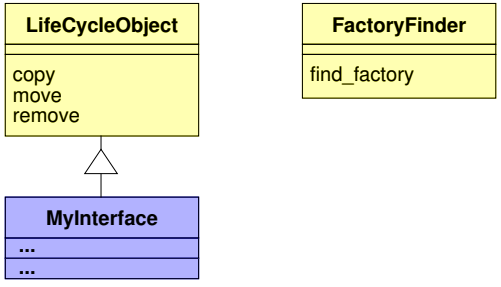


• With the call of remove the object is deleted.

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Life Cycle Service (3)

■ Objects need to implement the LifeCycleObject interface



◆ copy and move need a FactoryFinder object to find a factory object which in turn will create the copy or migrated object

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Life Cycle Service (4)

■ Example: implementation of the copy method
- ◆ Client invokes copy method and passes a reference to a FactoryFinder
- ◆ The copy method is provided by the programmer **or** by the CORBA implementation
- ◆ The copy method calls the FactoryFinder and selects a suitable factory that is used to create a new object
- ◆ The new object will be initialized with the data of the existing object

▲ To the outside: clearly defined interface

▲ To the inside: open and implementation dependent, e.g., protocol between copy method and factory

■ In CORBA 2.0 relationship between objects is considered for copying and mobility: part-of relationship, etc.
- ◆ Parts of the object are also copied or migrated (deep copy vs. shallow copy)

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.