



Architecture of the CORBA Component Model

CORBA 3.0



What is CORBA

CORBA (Common Request Broker Architecture) is a distributed object-oriented client server platform.

It provides:

- ✦ An object oriented remote procedure call mechanism (RPC)
- ✦ Object Services (Naming, Trading)
- ✦ Language mappings
- ✦ Interoperability protocols
- ✦ Programming conventions and design patterns

CORBA replaces ad-hoc special mechanisms (e.g. socket communication) with an open, standardized and portable platform.



The Object Management Group (OMG)

- ✦ Founded 1989, Head-quarters in Framingham, MA
- ✦ 2001: 167 contributing members (71+59+37)
 - 160 influencing, 117 university, 197 auditing, 14 government
- ✦ CORBA: Common Request Broker Architecture
 - CORBA 1.0 (1991): Object Request Broker, IDL, C
 - CORBA 2.0 (1995): Interoperability, C++, ...
 - CORBA 3.0 (2002): Components, Scripting, Real-Time
 - Domain Specifications
- ✦ UML: Unified Modelling Language

OMG Task Forces

- ✦ Board of Directors, Architecture Board, Platform Technical Committee, Domain Technical Committee

- ✦ PTC Task Forces:

- Object Request Broker/Object Services (orbos)
- Analysis and Design (ad)

- ✦ DTC Task Forces:

- Business Objects (bodtf)
- Electronic Commerce (ec)
- Finance
- Health Care
- Life Sciences Research

- Manufacturing (mfg)
- Telecommunications
- Transportation
- Utilities

- ✦ Special Interest Groups:

- Benchmarking
- Realtime
- Japan
- Security
- Distributed Simulation
- C4I
- ...



What is Client-Server-Computing?

- ✚ A group of clients and servers cooperate in solving a problem
- ✚ Servers are passive participants, providing a service, and waiting for requests from clients
- ✚ Clients are active participants, requesting a service from the server.
- ✚ Clients and servers run (typically) as operating system processes on different computers
- ✚ Object-oriented client-server computing adds OO aspects: interfaces, messages, inheritance, polymorphism.

Advantages and disadvantages of CORBA

Advantages:

- ✖ Implementations based on vendor-independent and open standards, available on a multitude of hardware platforms, operating systems, and programming languages
- ✖ Disencumbers of the slavish tasks in distributed computing

Disadvantages:

- ✖ No reference implementation
- ✖ Defined by consensus and compromise
- ✖ Not perfect
- ✖ „can shoot yourself in the foot and blow the whole leg off“



Heterogeneity

CORBA works for homogenous and heterogenous environments.

Characteristics of heterogenous environments are:

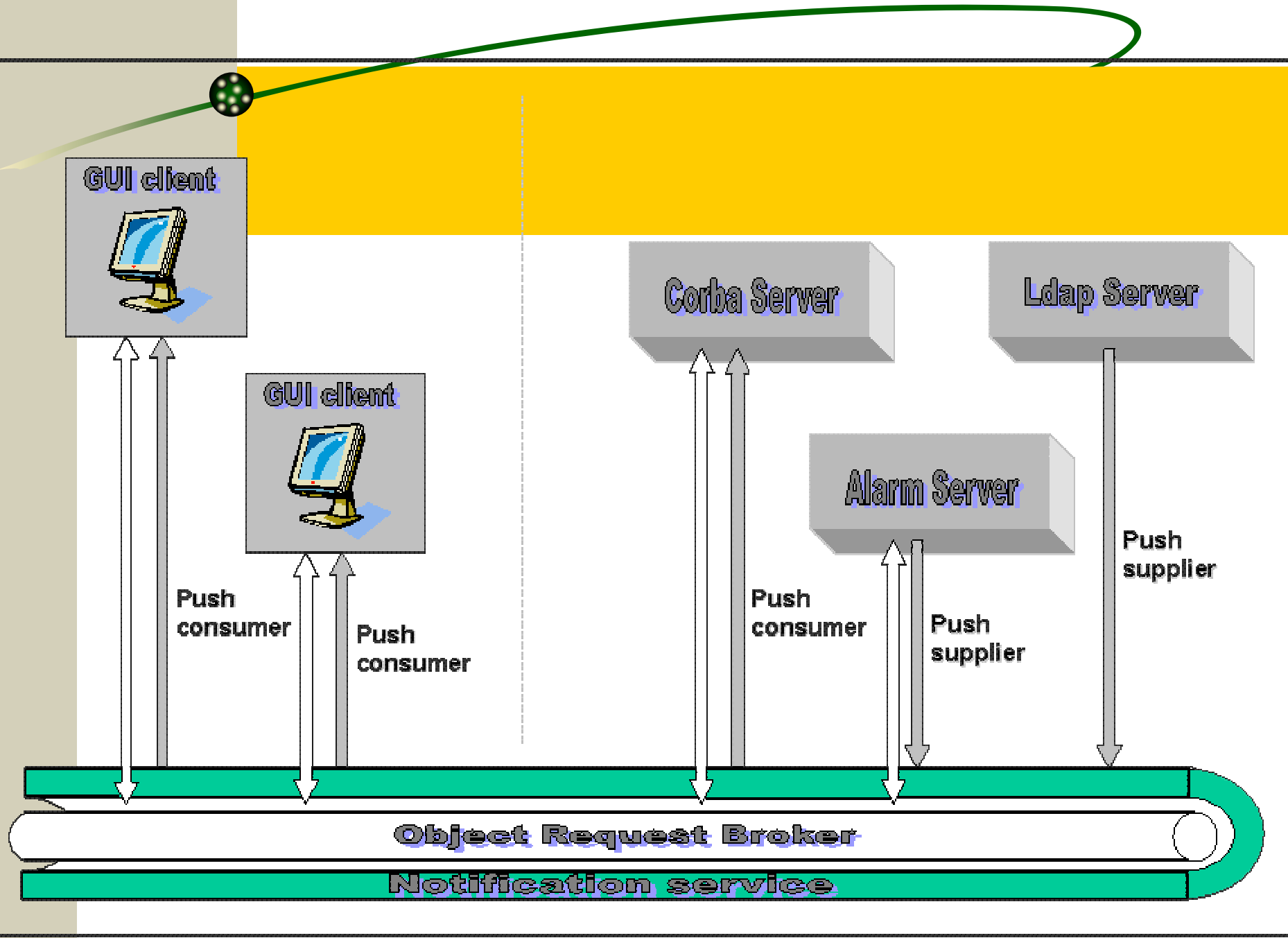
- Language transparency
- Location transparency
- Service transparency
- Implementation transparency
- Architecture transparency
- Operating system transparency
- (Protocol transparency)
- (Transport transparency)



Examples (1)

Alcatel OmniVista

- ✦ Private Branch Exchange (PBX) Network Management
- ✦ Usage of the Notification Service
 - Distribution of alarms
 - Notification about changes in the LDAP repository



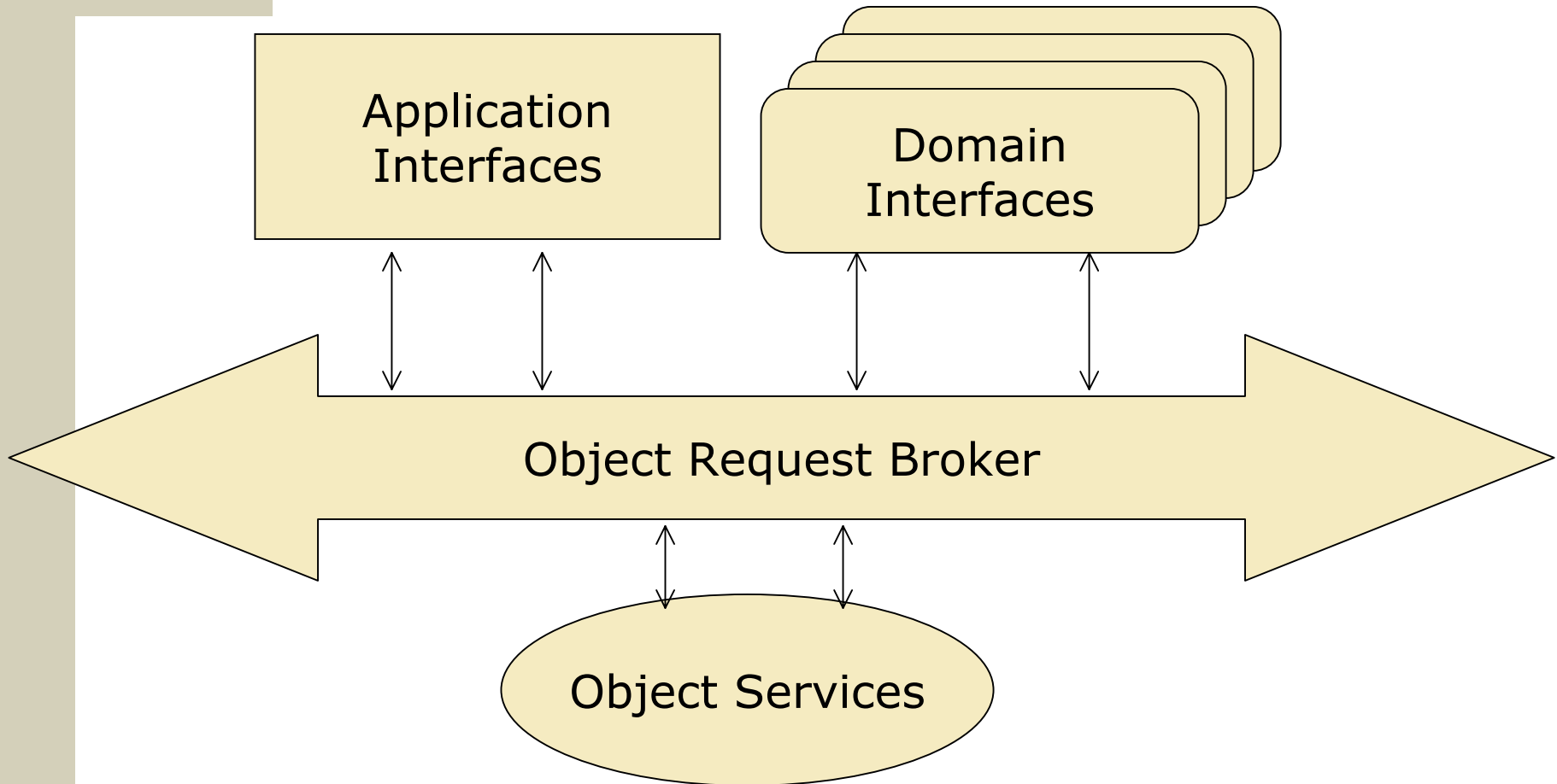


Examples (2)

Nokia Intelligent Network

- ✦ Intelligent Network: special services, provided by network or customer
 - 0130/0180/0190
 - televoting
 - number portability
- ✦ Nokia: Service development using CORBA

Object Management Architecture (OMA)





Objects

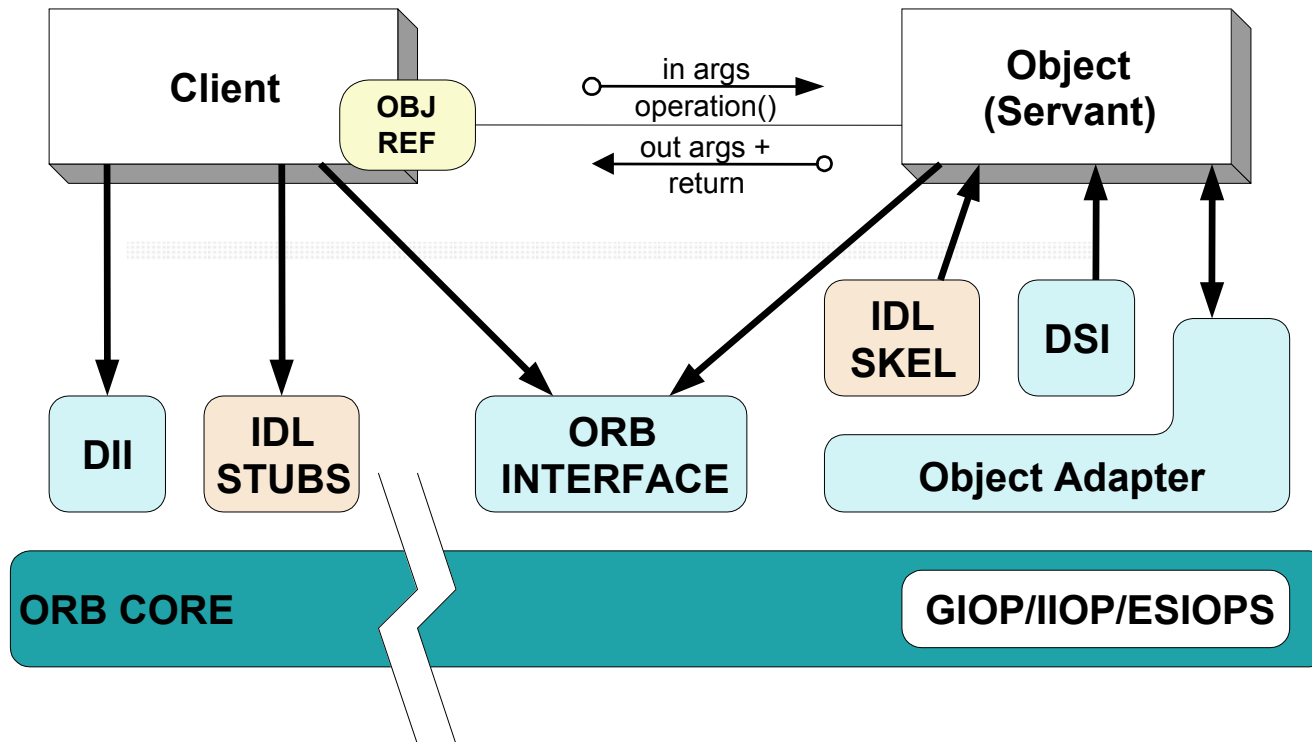
- ✚ Object: encapsulated entity with immutable specific identity, interacts only through well-specified interfaces
- ✚ CORBA object \neq Java object (language-specific object)
 - A single Java object can implement multiple CORBA objects
 - A CORBA object can successively be implemented (incarnated) through multiple Java objects
 - Most simple case: one-to-one relationship between CORBA objects and Java objects (servants)



Interfaces

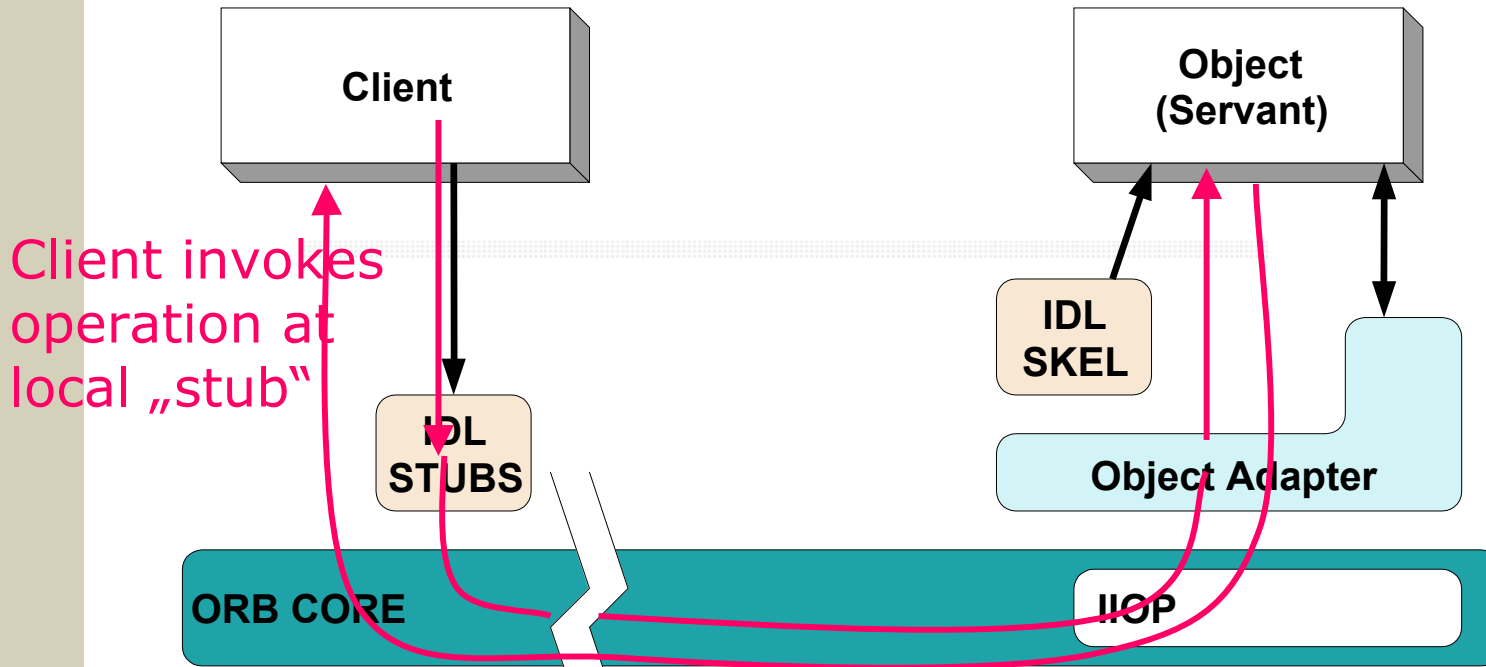
- ✚ Interface: Set of operations
 - Object services: Interfaces required in many applications independently from application domain
 - Horizontal services, infrastructure services
 - Domain interfaces: standardized interfaces for services specific to an application area
 - Vertical services
 - Application interfaces: specific for the application, not standardized
- ✚ Interfaces are defined using IDL (Interface Definition Language)

ORB: Object Request Broker



Method Call

The result is sent back



Client invokes operation at local „stub“

Local ORB encodes parameters

Remote ORB decodes



Method call: Control flow

Client sends a request:

- ✚ Client ORB analyses object reference
- ✚ Connects to server if necessary
- ✚ Sends request: (target object, operation name, parameters)

ORB receives request:

- ✚ Activates server (if no server active)
- ✚ Activates target servant (if no servant active)
- ✚ Invokes method, waits for completion
- ✚ Sends reply (if required)

Method semantics:

- ✚ At-most-once: exception in case of error
- ✚ Best effort (oneway)



Interoperability

- ✦ CDR: Common Data Representation
- ✦ GIOP: General Inter-ORB Protocol
- ✦ IIOP: Internet Inter-ORB Protocol
- ✦ Alternative protocols:
 - ESIOP: Environment-specific Inter-ORB protocols
 - DCE-CIOP
 - SCCP (Signalling Control Part) IOP
 - Pluggable Protocols

Object references

- ✚ Every reference identifies exactly one object
- ✚ Different references may refer to the same object
- ✚ References may be „nil“
- ✚ References may become invalid (*dangle*): The object referred-to has already disappeared
- ✚ References are opaque
 - Note: „Interoperable Object References“ (IOR)
- ✚ References are strongly typed.
- ✚ References allow late binding.
- ✚ References may be persistent.

Object services

- ✚ Naming Service (CosNaming)
- ✚ Trader (CosTrading)
- ✚ Property Service (CosPropertyService)
- ✚ Event Service (CosEventComm,
CosEventChannelAdmin)
- ✚ Notification Service (CosNotification)
- ✚ ...
- ✚ (Transactions)
- ✚ (Persistency) (PSS: Persistent State Service)