Middleware: Challenges and Evolution from Procedural to Service Orientation

Bruno Traverson (bruno.traverson@edf.fr)

Middleware

[Meeting in the Middle, Ruth Palmer]

- Contraction of « middle » and « software ».
- Définition
  - Core software independent of any application domain and generally covering four categories of functionalities:
    - Communication,
    - Coordination,
    - Conversion,
    - Facilitation.

[Mehta, Medvidovic, Phadke, Towards a taxonomy of software connectors. Software Engineering, 2000]

Software Architecture

[Constantine, Farid Benyaa]

- Definition
  - Structure of a system and its components, their relationships and the mainlines that guide their design and their evolution during time.

- Four main approaches in middleware
  - Procedural,
  - Component-based,
  - Object-oriented,
  - Service-oriented.

Contents

- Basic principles
- Procedural approach
- Object-oriented approach
- Component-based approach
- Service-oriented approach
- Conclusion

Contents

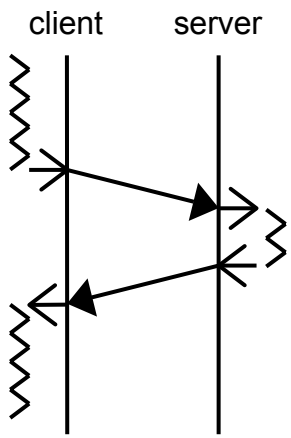
- Basic principles
- Procedural approach
- Object-oriented approach
- Component-based approach
- Service-oriented approach
- Conclusion

Communication

- Communication functions insure exchanges among the various elements of a distributed application.
- Classification based on the kind of the flow
  - Information flow (data or code),
  - Control flow (signal or call).
- And on the kind of binding between elements
  - Direct binding (elements know each other),
  - Indirect binding (elements communicate via a tier).

Client/Server

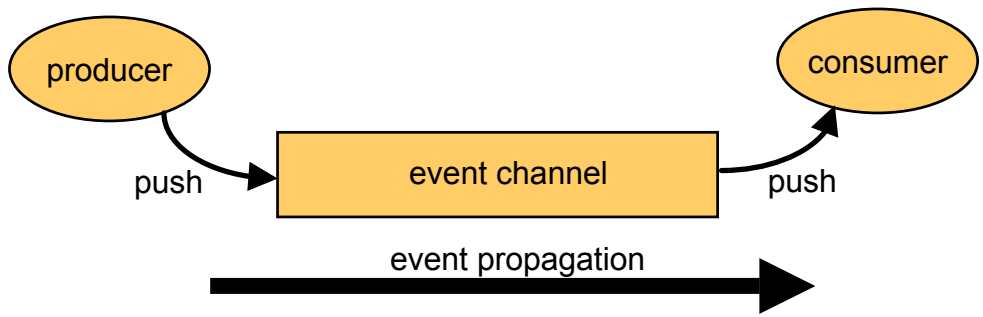
- One entity playing the « client » role calls an operation offered by another entity playing the « server » role.



- Characteristics
  - Control flow with direct binding
  - Simultaneous presence of client and server entities,
  - Blocking call for the client.

Producer/Consumer

- Entities playing the « producer » role emit events to other entities playing the « consumer » role.
- Event producers and consumers communicate through a tier called « event channel » (anonymous communication).



- Characteristics
  - Information flow with indirect binding,
  - Muti-peer communication,
  - Push or Pull model.

Coordination

- Coordination functions organize the global execution of the various elements of a distributed application.
- Classification based on the coordination strategy
  - Based on concurrency between activities (scheduling functions : serialization and causality orders, transaction)
  - Based on cooperation among activities (consensus algorithms : distributed termination, workflow).

Transaction

- Definition
  - A set of operations that makes the Information System go from one initial consistent state to another final consistent state.
- Corollary
  - Intermediate states may be inconsistent.
- Four ACID properties
  - Atomicity
  - Consistency
  - Isolation
  - Durability

Workflow

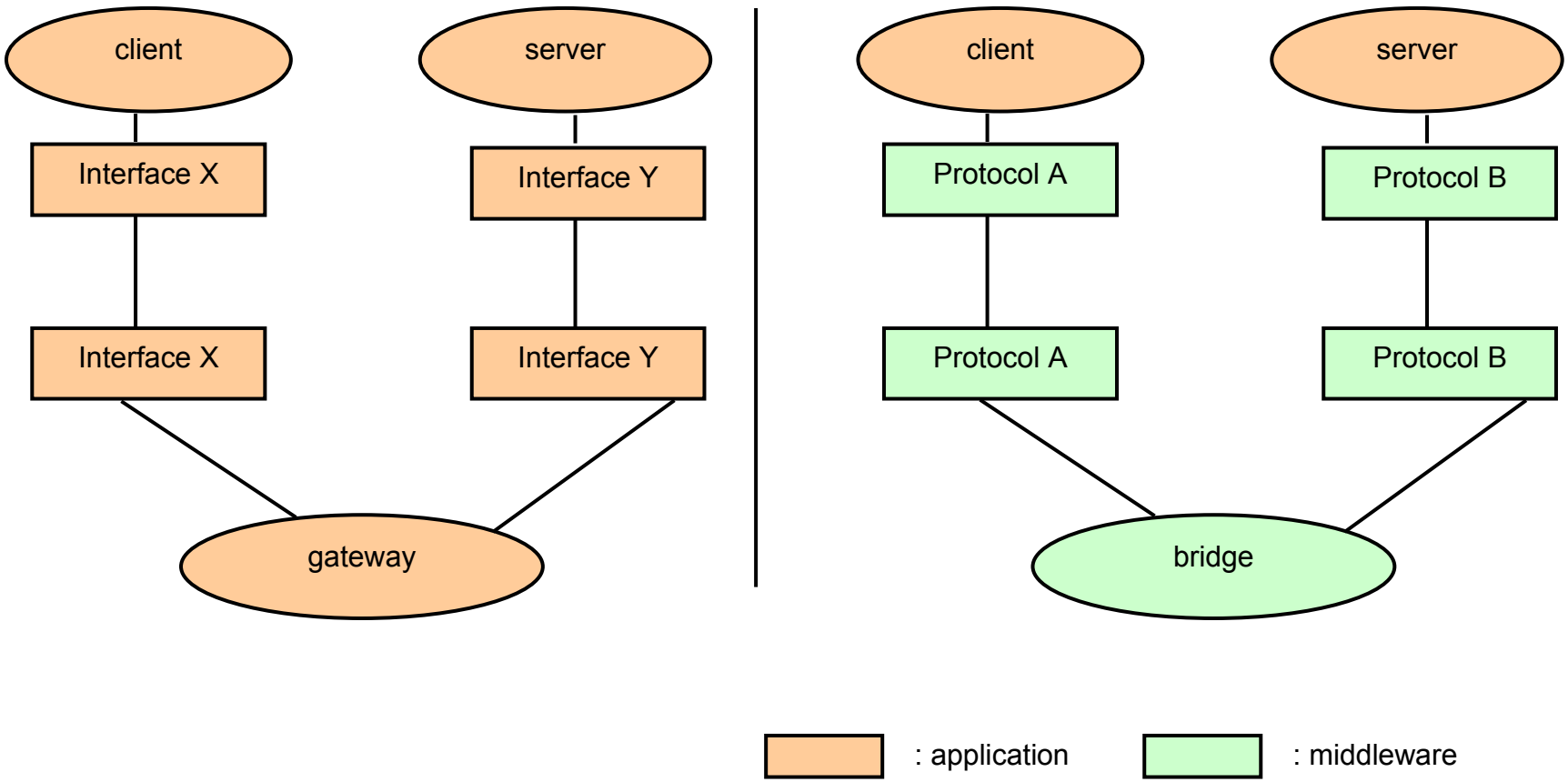
- Limits of transaction management
  - A transaction cannot commit if one of its operations fails (atomicity),
  - A transaction cannot cooperate with other transactions (isolation),
  - A transaction must be short-lived (atomicity and isolation).
- Definition

  A set of "open-ended" operations (uncertain duration, conditional execution, potential interactions with other activities).

Conversion

- Conversion functions mask the disparities among the various elements of a distributed application.
- Classification based on the level of conversion
  - Interface adaptation,
  - Communication protocol,
  - General model.
- And the responsibility of the conversion
  - Application responsibility (gateway),
  - Middleware responsibility (bridge),
  - Shared responsibility (context-awareness, pervasive systems).

Basic Conversion Techniques



Adaptable Middleware

- Modification of all the system or of one of its parts in a transparent way with regards to its operational state in order to satisfy a particular concern (error correction, performance optimization).
- Characteristics
  - Manual adaptation
    - Adaptation strategy handled by a human administrator,
    - Event propagation when evolution is detected.
  - Automatic adaptation
    - Self-adaptation following registered adaptation strategies,
    - Reflection mechanism allowing self-observation and self-modification.

Facilitation

- Facilitation functions optimize interactions among the various elements of a distributed application.
- The main goal is to enhance performance and reliability characteristics of the application or, more generally, to manage the quality of service (QoS).
  - Fault tolerance,
  - Load balancing.

QoS Contracts

- Definition
  - Interval of tolerance between QoS requests and offers (in case of environment failure, offer a service of lower quality but still acceptable by the user).
- Characteristics
    - Composition,
    - Observation,
    - Guarantee,
    - Negotiation.

Contents

- Basic principles
- Procedural approach
- Object-oriented approach
- Component-based approach
- Service-oriented approach
- Conclusion

Procedural Approach

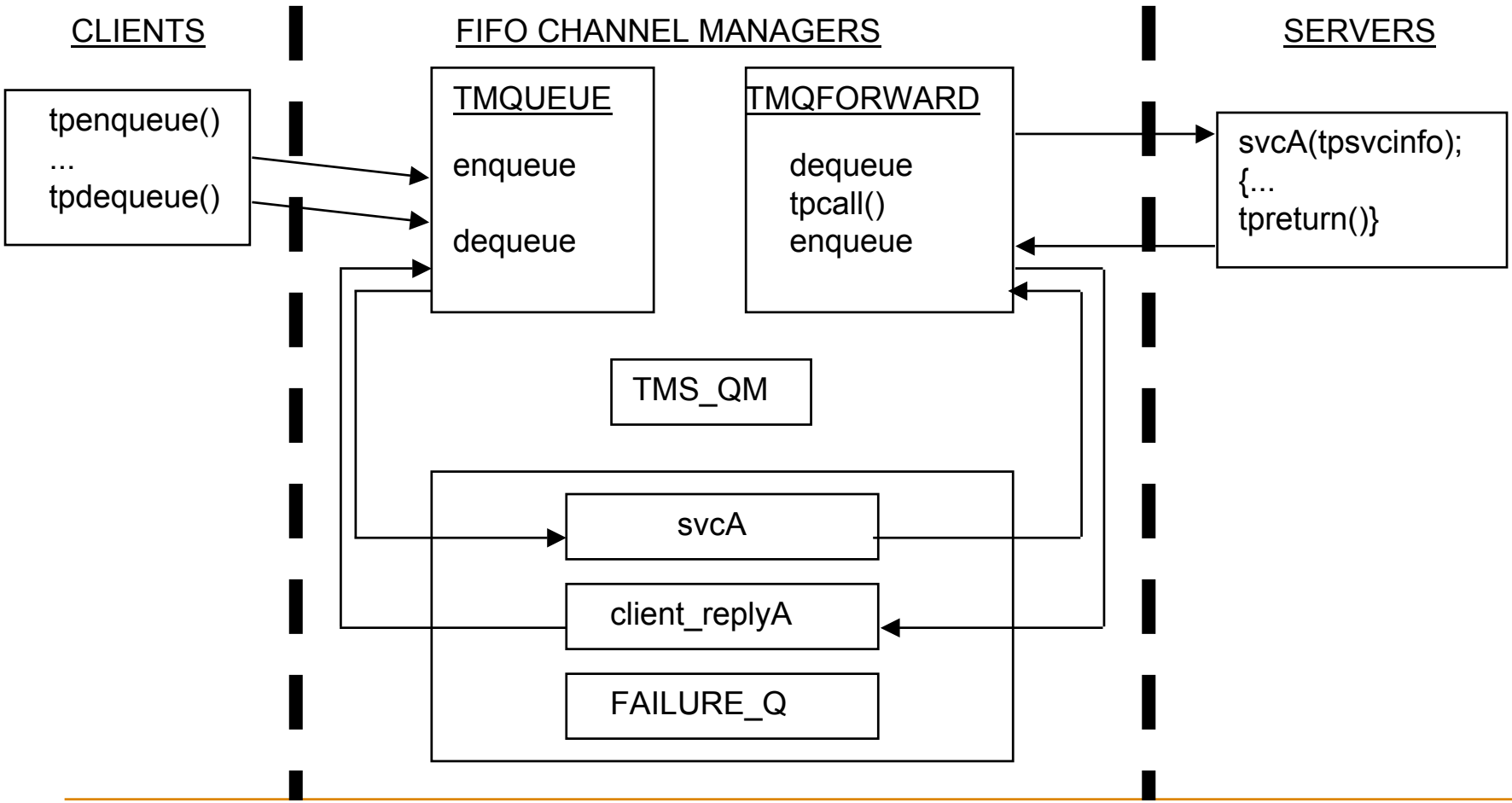
- The system is splitted into functional modules. When the modules are running on remote machines, middleware supports distribution.
- Main standards
  - DCE : Distributed Computing Environment,
  - DTP : Distributed Transaction Processing.
- Main realizations
  - Distributed Operating Systems,
  - Distributed Transaction Processing Monitors.

Communication Paradigms

- Asynchronous Processing
  - Entities communicate through a tier,
  - Exchange flow is one-way.
- Conversational
  - Entities communicate as peers,
  - Information is exchanged inside a session with respect to control rules defined by peers.
- Request/Response
  - One entity calls the other,
  - Exchange flow is two-way.

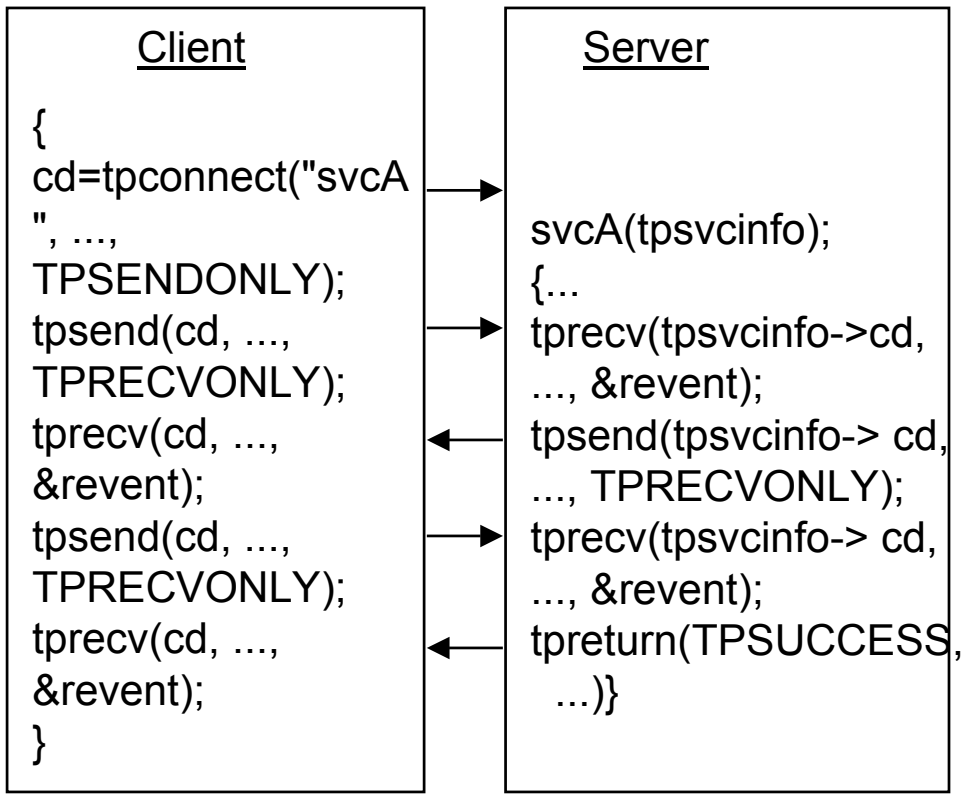
Asynchronous Processing

- Using a Programmatic Interface



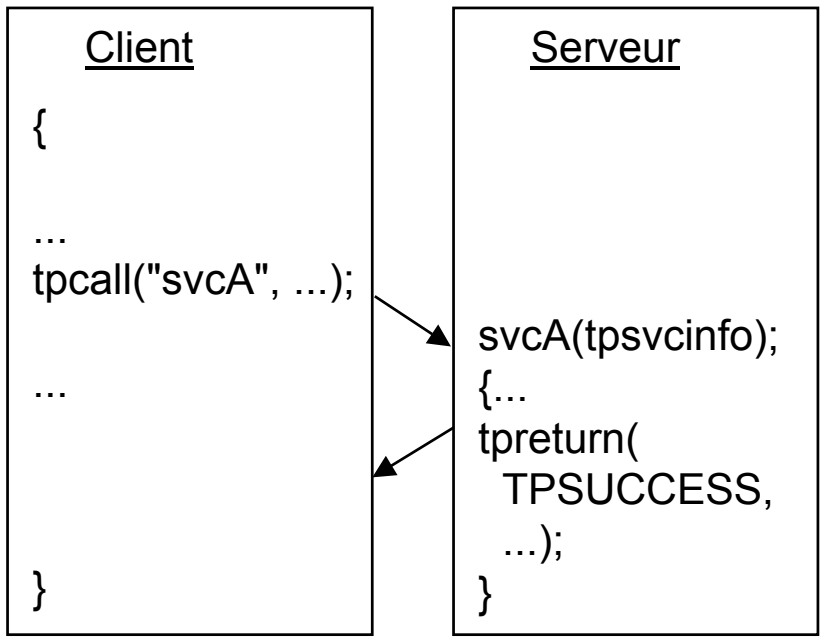
Conversational

- Using a Programmatic Interface



Request/Response (1)

- Using a Programmatic Interface



Request/Response (2)

- Using a Language Extension
  - The interface offering remote operations is defined using a language called IDL (Interface Definition Language).
  - Communication between client and server involves adaptators to network based on the interface definition.

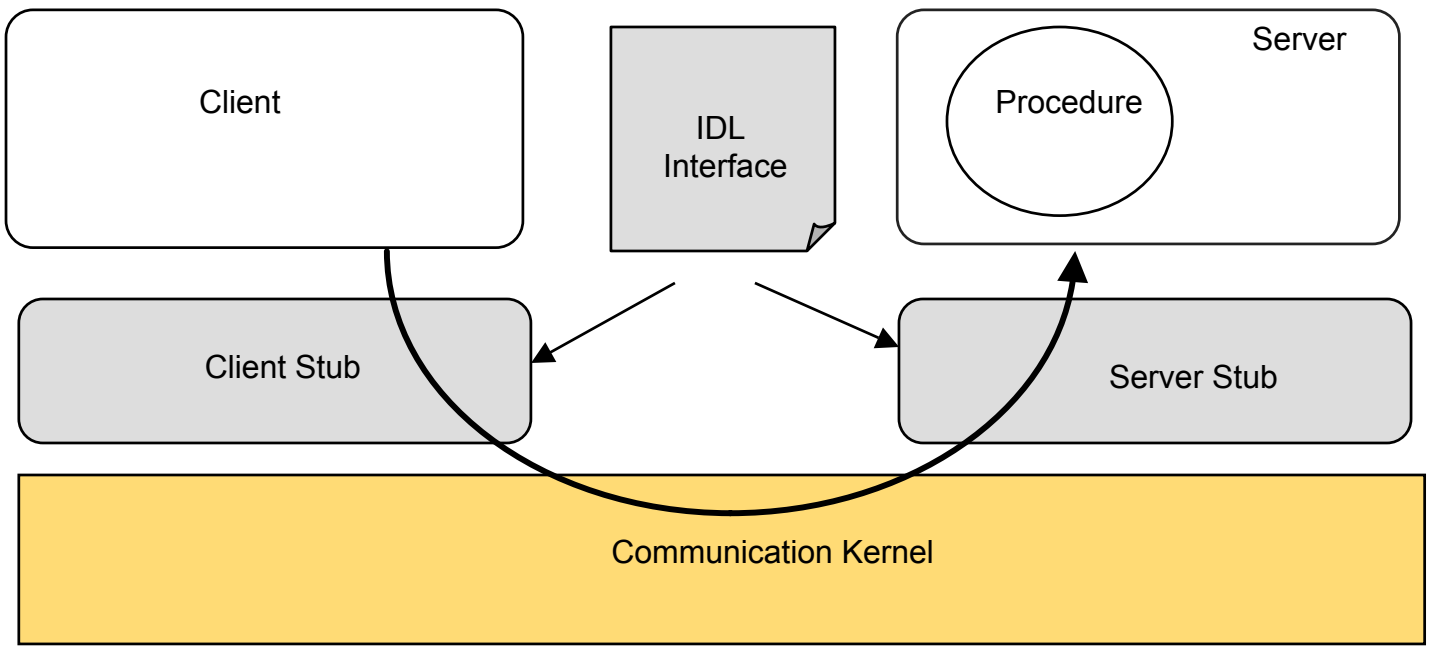
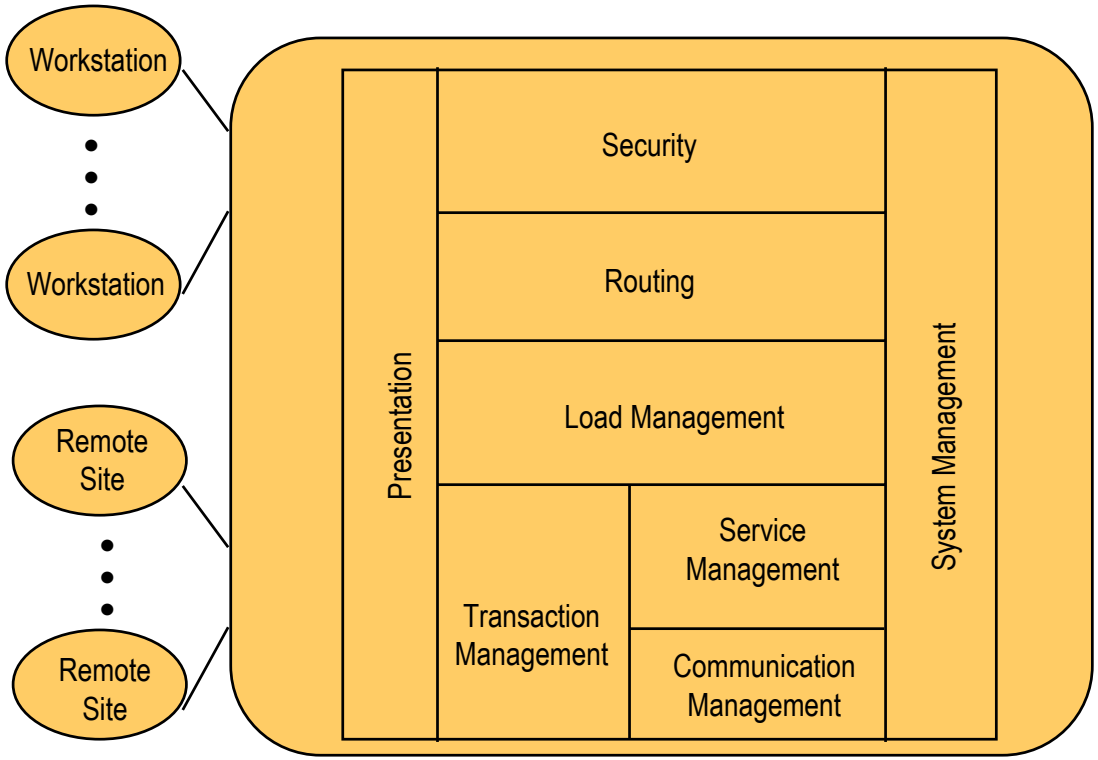


Illustration: TP Monitors



Main Functions (1)

- Security
  - The TP monitor must check that the service request from a user owns the right to be processed (authentication, access right lists, integrity and privacy management).
- Routing
  - Scheduling mechanism similar to that used in an operating system
    - Lookup for local availability then for remote availability,
    - Lookup for a local server instance,
    - If no instance, launching of a new server instance or queuing.
- Load management
  - Used if more than one server may satisfy the service request.
  - Two strategies
    - Minimize response times (load balancing),
    - Maximize system throughput (load sharing).

Main Functions (2)

- Transaction management
  - Used for commitment and recovery,
  - Atomic commitment, logging, recovery procedures.
- Service management
  - Ensures transactional execution of the service requests,
  - Transaction context management, service association to a server.
- Communication management
  - Ensures transactional communication,
  - Transaction context propagation, communication paradigm support (request/response, conversational, asynchronous processing).

Conclusion on Procedures

- Assets
  - Interface Definition Language (IDL),
  - Application Programmatic Interface (API).
- Advantages
  - “Natural” distribution of processing,
  - Transactional execution.
- Limits
  - Tightly-coupled systems,
  - Interoperability between domains still difficult.

Bibliography on Procedures

- DCE: Remote Procedure Call. X/Open CAE Specification. C309. X/Open Company Limited, 1994.
- DTP : Distributed Transaction Processing Reference Model, Version 3. G307. X/Open Company Limited, 1996.

Contents

- Basic principles
- Procedural approach
- Object-oriented approach
- Component-based approach
- Service-oriented approach
- Conclusion

Object-oriented Approach

- The system is viewed as interacting objects. When the objects are running on remote machines, middleware supports distribution.
- Main standards
  - CORBA : Common Object Request Broker Architecture,
  - COM : Common Object Model.
- Main realizations
  - Object Brokers.

Object

- Definition
  - Entity containing as well data (attributes) as processing (methods) accessible through an interface.
- Used in a lot of domains
  - Modeling methodologies,
  - Programming languages,
  - Databases,
  - Distributed environments.

Basic Principles

- Wrapping
  - Object description is separated from its realization.
- Composition
  - Multiple objects may be combined to form one object.
- Inheritance
  - Relationship between objects that permits to define new objects by using characteristics of existing objects.

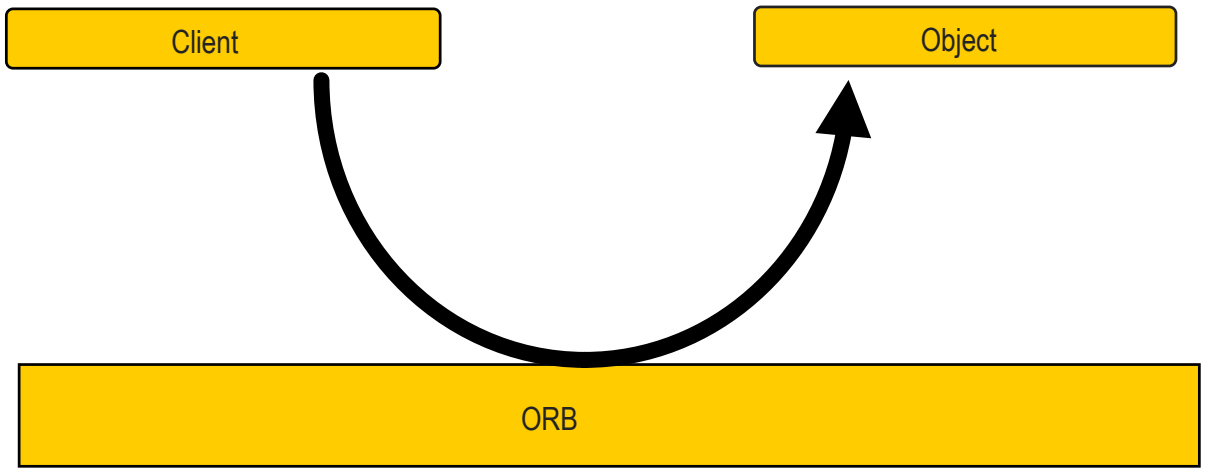
Assets

- Evolution support
  - Replacement or modification of a component with no impact on its environment (wrapping).
- Legacy integration
  - Mixing of existing and new applications (wrapping and composition).
- Extension support
  - New components may be built from existing objects (composition and inheritance).

Illustration : CORBA

- OMG: Object Management Group
  - International consortium composed of hardware manufacturers, software editors and users.
  - Produce specifications for the design of distributed object-oriented software applications.
- OMG specifications
  - CORBA (Common Object Request Broker Architecture)
  - OMA (Object Management Architecture)

CORBA Architecture (1)

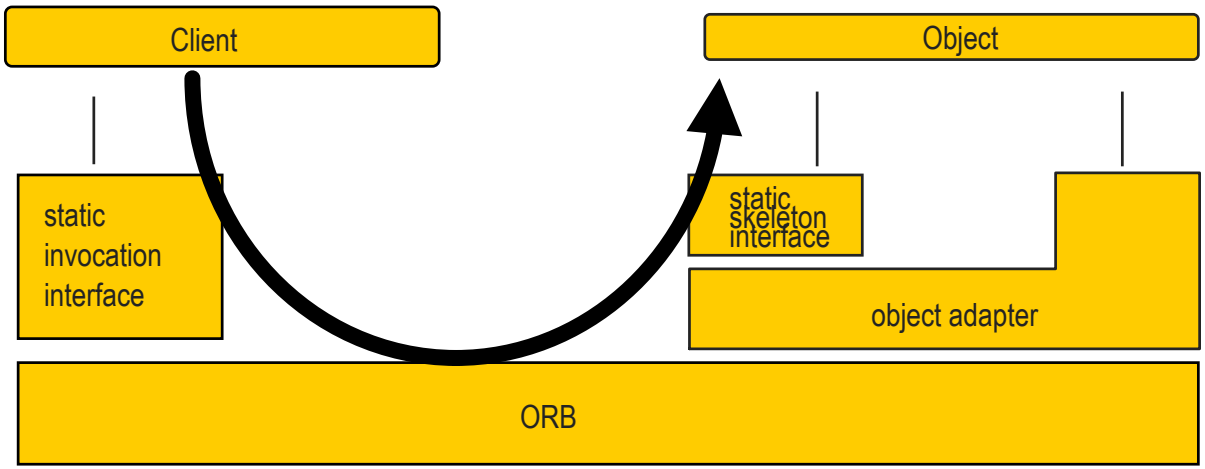


- ORB: Object Request Broker
  - Vehicle for requests between objects.

ORB Main Components

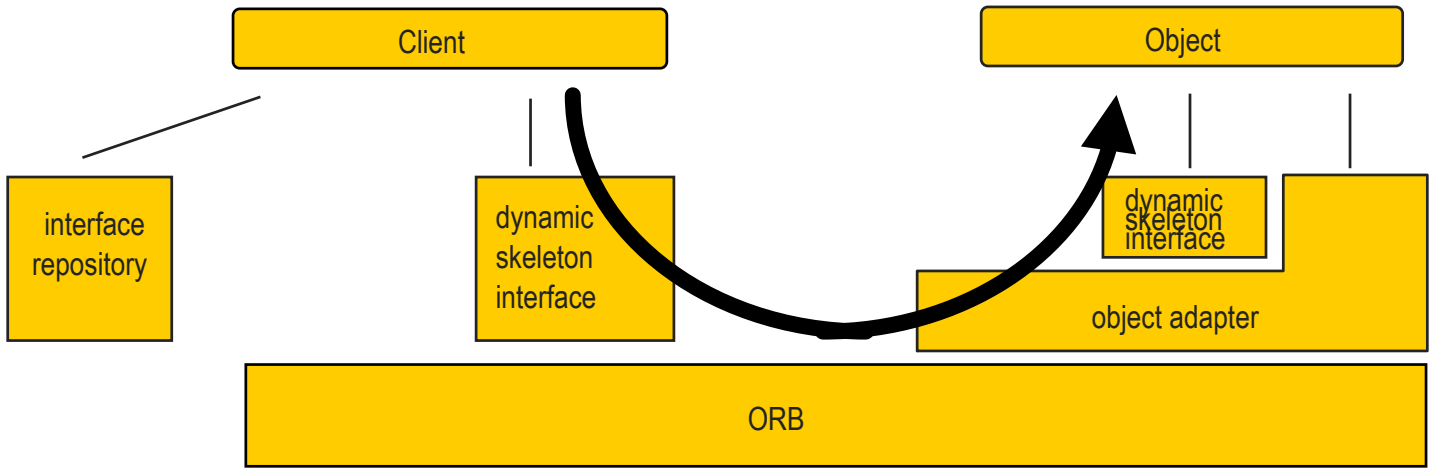
- IDL : Interface Definition Language
  - Language used to describe object interfaces.
  - Related to portability support.
- IIOP : Internet Inter-ORB Protocol
  - Request/response communication paradigm.
  - Related to interoperability support.

CORBA Architecture (2)



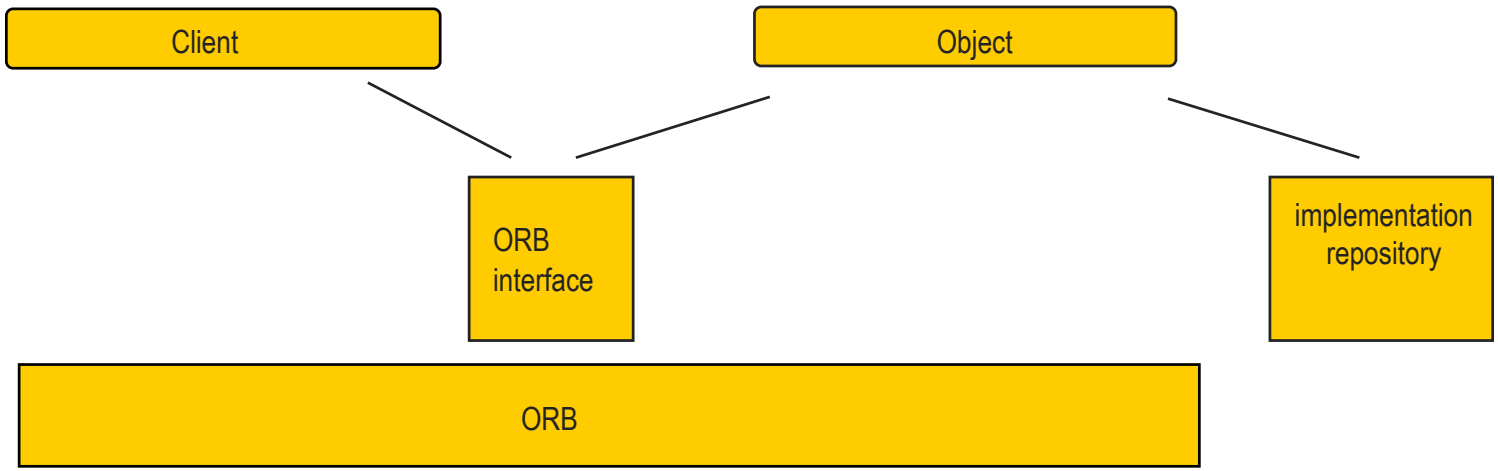
- Static invocation
  - Client and server interfaces that ensure marshalling/unmarshalling functions related to object interactions.
- Object adapter
  - Routing function in the server side,
  - i.e. Establish the link between CORBA object references and implementation object references (servants).

CORBA Architecture (3)



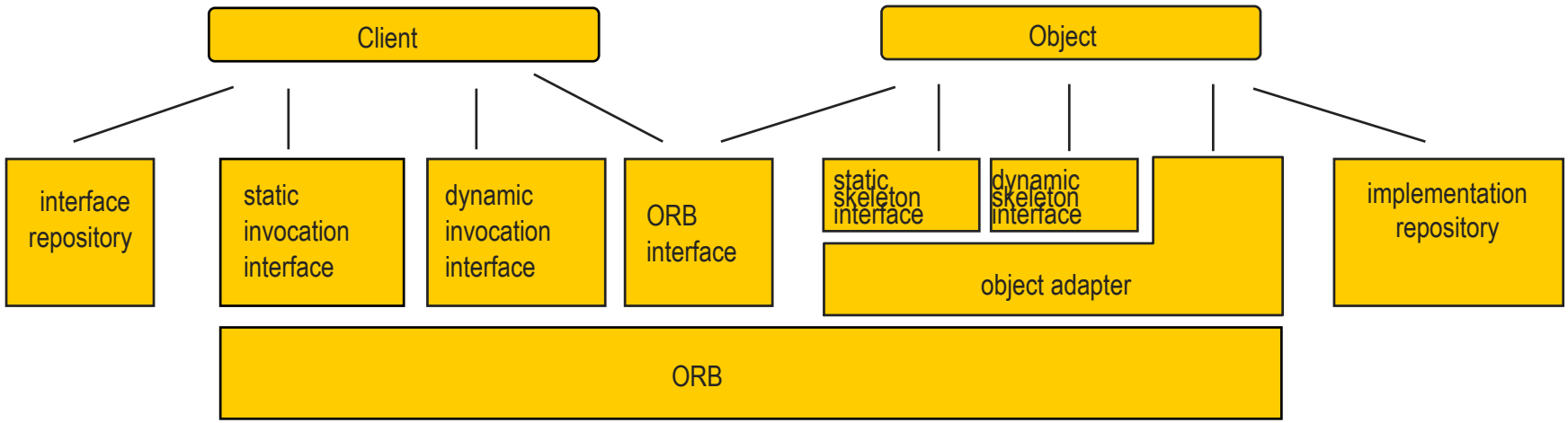
- Dynamic invocation
  - Permits to send and to receive requests on objects unknown at compile time.
- Interface repository
  - Contains IDL definitions.

CORBA Architecture (4)



- ORB interface
  - Services for object reference management and initialization.
- Implementation repository
  - Contains object implementations.

CORBA Architecture (5)



- A global view ...

Conclusion on Objects

- Assets
  - Unification of information and control flow.
- Advantages
  - Evolutivity and extensibility,
  - Standardization of horizontal functions (services) and vertical functions (facilities, domain objects).
- Limits
  - Tightly-coupled systems,
  - Programmation and management still complex.

Bibliography on Objects

- OMG, The Common Object Request Broker: Architecture and Specification. Version 3.0, July 2002.
- M. Henning, S. Vinoski, *Advanced CORBA Programming with C++*, Addison-Wesley, 1999.

Contents

- Basic principles
- Procedural approach
- Object-oriented approach
- Component-based approach
- Service-oriented approach
- Conclusion

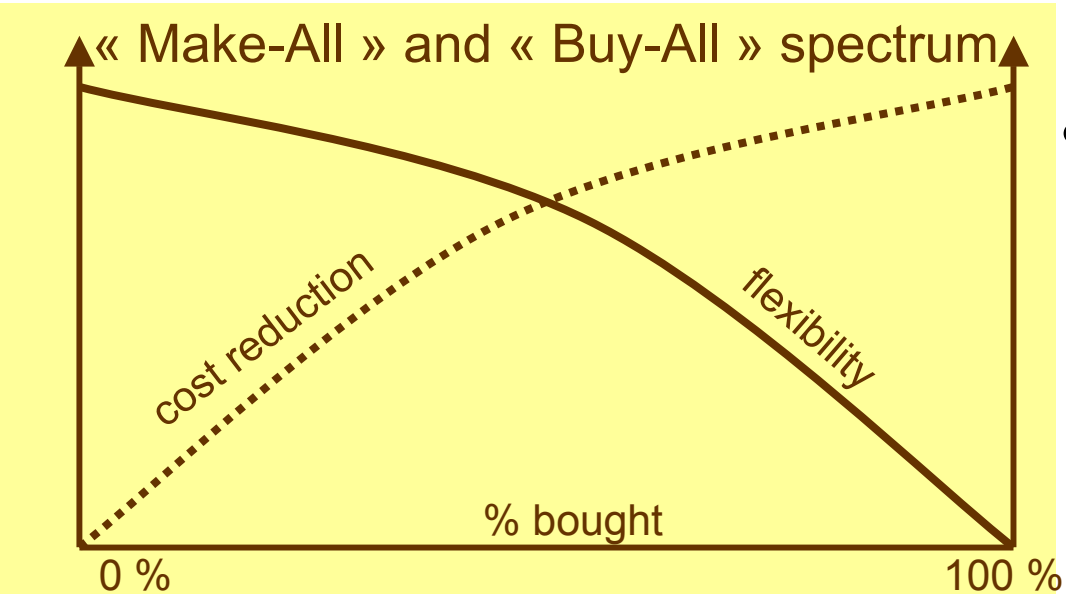
Component-based Approach

- The system is the result of the assembly of its components. When the components are running on remote machines, middleware supports distribution.
- Main standards
  - CCM : CORBA Component Model,
  - EJB : Enterprise JavaBeans,
  - COM+ : Common Object Model.
- Main realizations
  - Application Servers.

Component (1)

- Software development may vary between two options
  - Specific software programming
    + Satisfaction of user requirements,
    - Expensive realization and maintenance,
    - Long time-to-market.
  - Generic software tuning
    + Cost and risk decrease,
    - Adaptation to user requirements harder,
    - Loss of local control.
- Software component approach is a compromise between these two options.

Component (2)



- Characteristics
  - Configuration,
  - Reuse,
  - Assembly.

Source : Component Software, Clemens Szyperski, Addison-Wesley

Configuration

- A component is designed and developed independently of applications that will use it.
- Adaptation of a component to its usage environment is called tuning or configuration
  - Application tuning,
  - System tuning.

Reuse

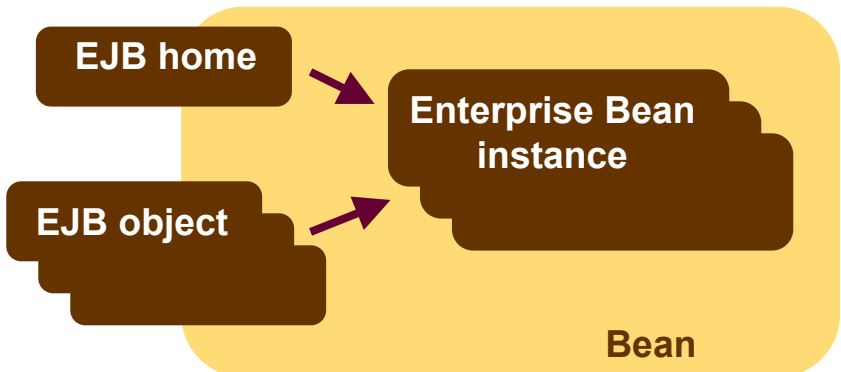
- Reuse permits the decrease in costs of realization and maintenance.
- Reuse implies strict design rules (design for and by reuse) and an adequate organization (reusable component repository).

Assembly

- Assembly enables effective usage of a software application in a given execution environment
  - At the level of a component: building constraints including what is required by the component.
  - At the level of a set of components: combine constraints including binding descriptions.

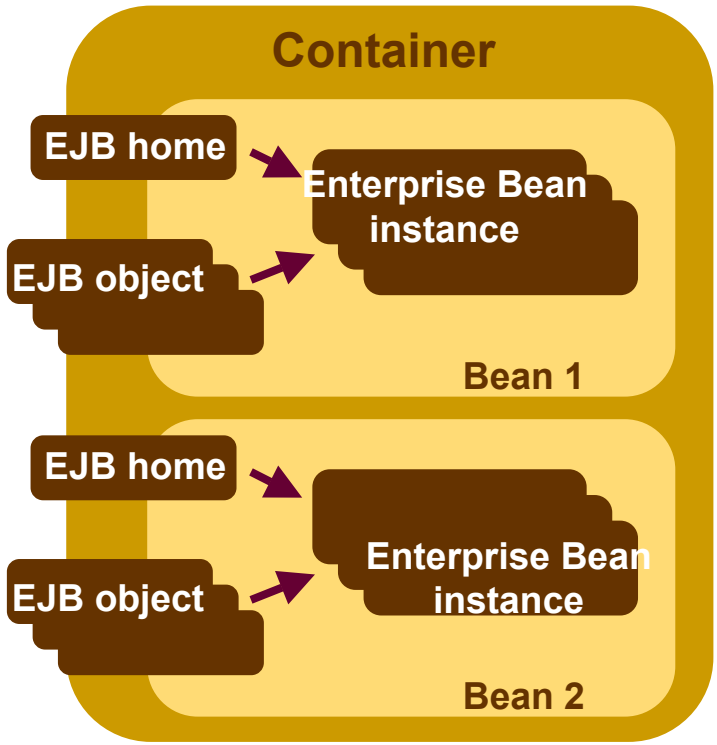
Illustration : Enterprise JavaBean

- Java component composed of two interfaces: EJBhome and EJBobject et one class: EnterpriseBean.
  - EJBhome corresponds to factory functions,
  - EJBobject represents the component interface,
  - EnterpriseBean constitutes the component implementation.



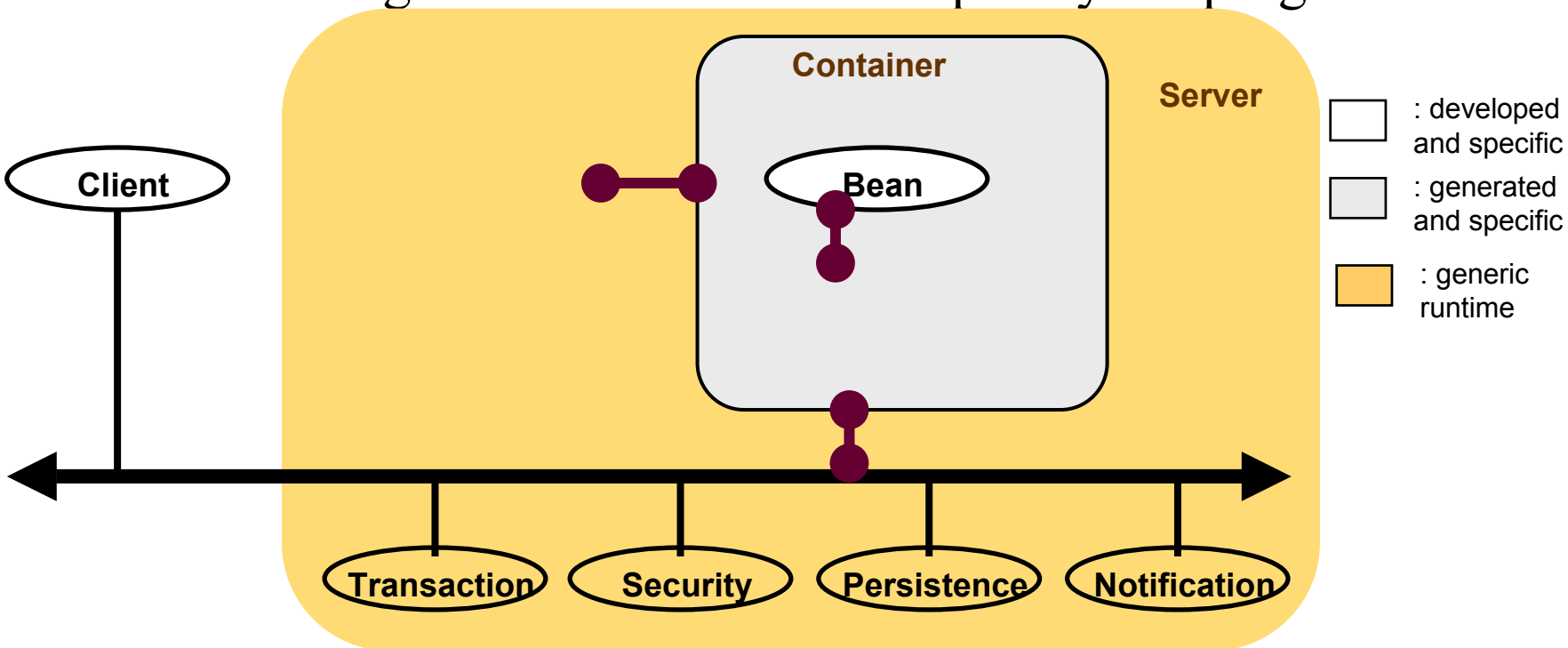
Container

- Set of Java interfaces and classes related to one category of component and linking the component and its hosting server.
- The component requirements in terms of environment are described in an XML deployment descriptor.



Server

- Execution environment fore one or more containers.
- Access to transaction, security, persistence and notification services.
- The server is generic and is not developed by the programmer.



Component Categories

- Session : component dedicated to one client
  - Without client context management (stateless session bean),
  - With client context management (stateful session bean).
- Entity : component shared among multiple clients
  - Managing by itself its persistence (bean-managed persistence),
  - Persistence management is delegated to container (container-managed persistence).
- Message-driven : component that is triggered by message receipts.

Conclusion on Components

- Assets
  - Standard containers.
- Advantages
  - Better separation between functional and technical concerns,
  - Declarative approach (descriptors).
- Limits
  - Limited choice of supporting functions and of lifecycle policies,
  - Some architectures are restricted to one programming language (EJB) or one operating system (COM+).

Bibliography on Components

- “ Enterprise JavaBeansTM Specification, Version 2.1 ”. Sun Microsystems, 2003.
- Clemens SZYPERSKI, « Component Software – Beyond Object-Oriented Programming ». Addison-Wesley, 1998.

Contents

- Basic principles
- Procedural approach
- Object-oriented approach
- Component-based approach
- Service-oriented approach
- Conclusion

Service-oriented Approach

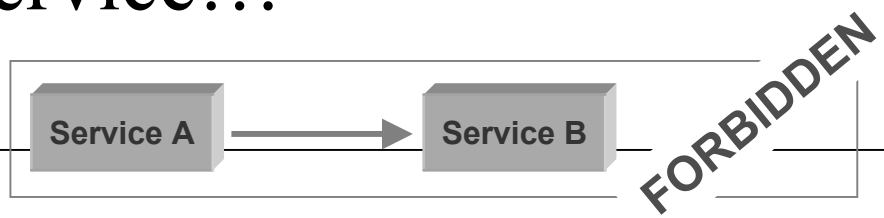
- The system is designed as a market of services with offers and demands that may match. When the services are running on remote machines, middleware supports distribution.
- Main standards
  - SOA: Service Oriented Architecture,
  - Web Services.
- Main realizations
  - Enterprise Service Bus.

Service Oriented Architecture

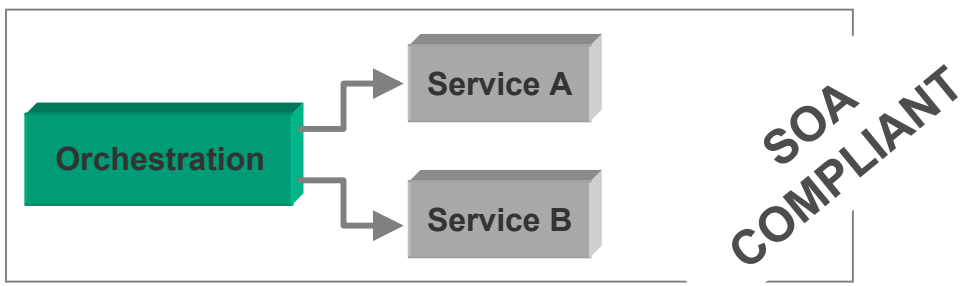
- Definition
  - Enables matching of service requests and offers.
- Characteristics
  - Take benefits of analogy with business services (easy mapping to organizational decomposition),
  - Externalize coordination functions (orchestration of services, loosely-coupling),
  - Better handle requester constraints (contract-based approach).

Loosely Coupling

- A service does not invoke directly another service…



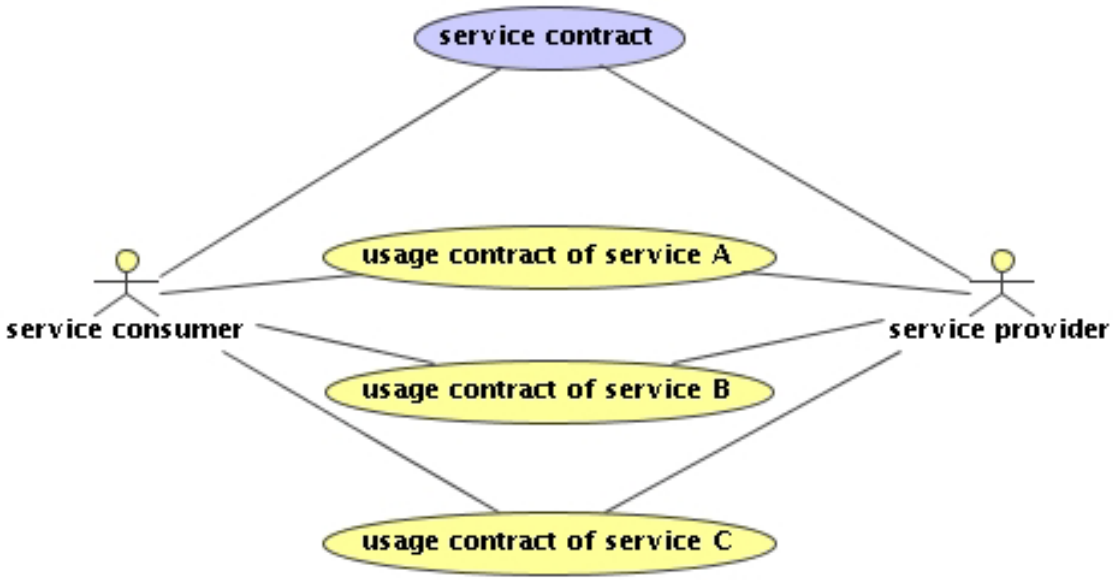
- …but delegates invocation to an external processing (orchestration).



+ Stateless

[Pierre Bonnet, Orchestra Networks]

Service Contract



- Collaboration between service provider and service consumer.

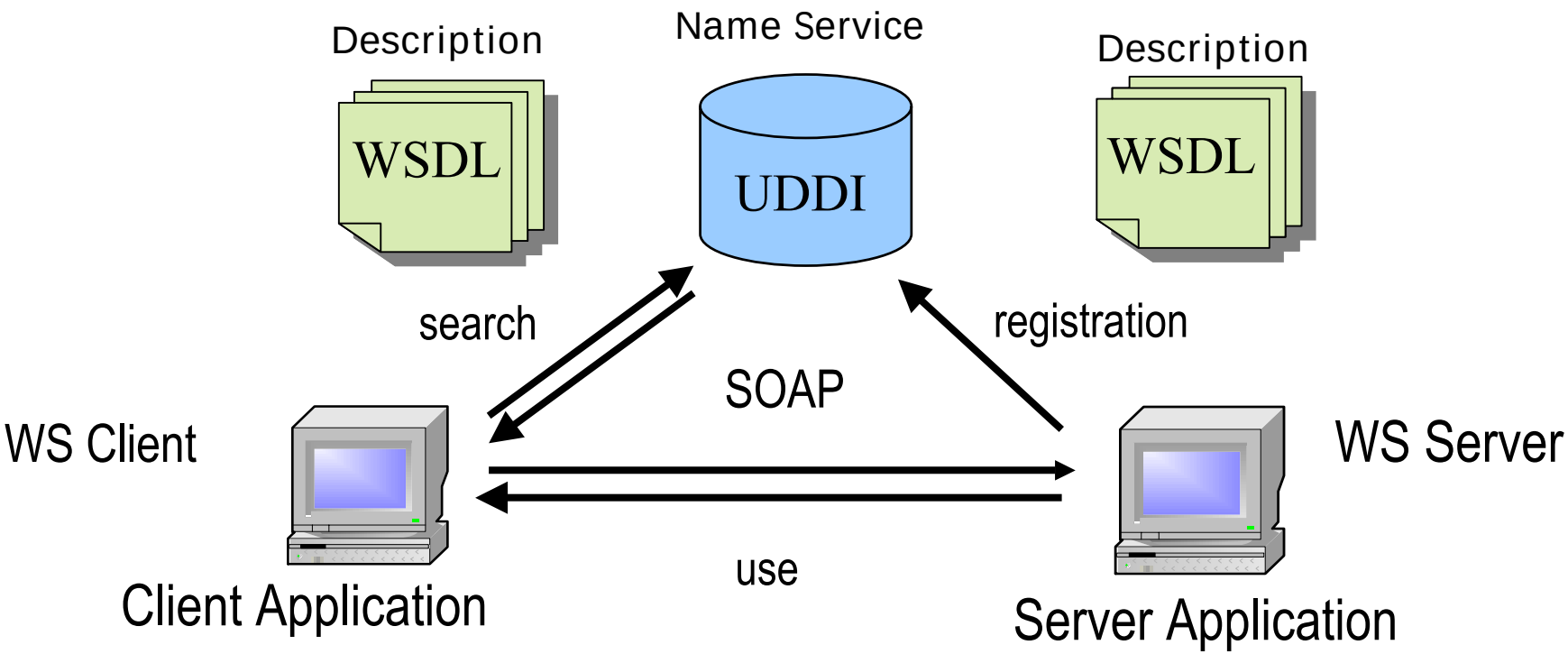
- Management of several usage contracts,
- Administration of variants and of versions.

[Pierre Bonnet, Orchestra Networks]

Illustration : Web Services

- In a first step, WebServices are used as **access solutions** to IS through Web techniques.
- Then, they should enable development of **simple interaction mechanisms** between applications through Internet.
- In the mid-term, scenarios more elaborated among several WebServices will imply **sophisticated cooperation mechanisms**.

Architecture (1)



SOAP : Simple Object Access Protocol

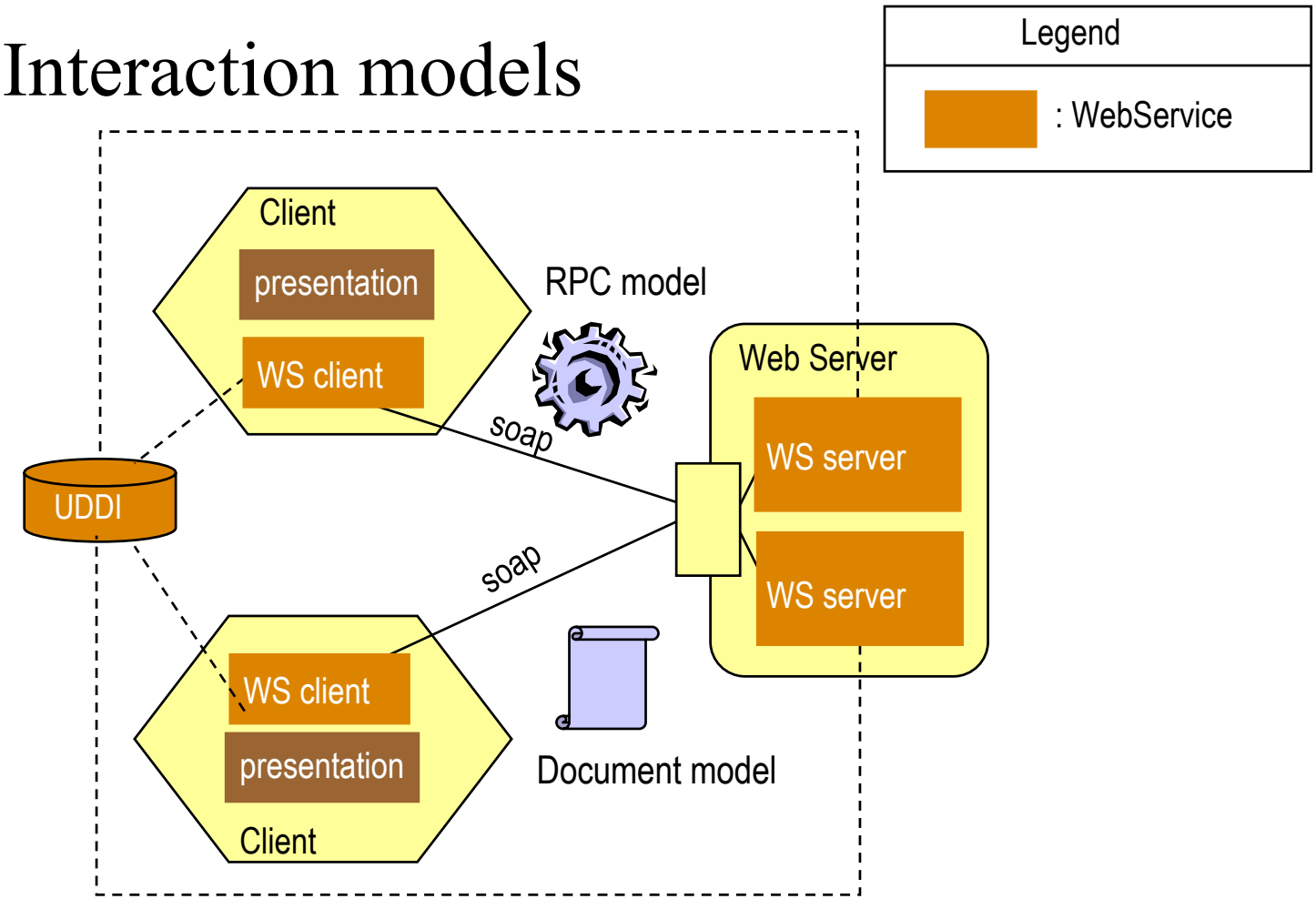
WSDL : Web Service Description Language

XML : eXtended Markup Language

UDDI : Universal Description Discovery and Integration

Architecture (2)

- Interaction models



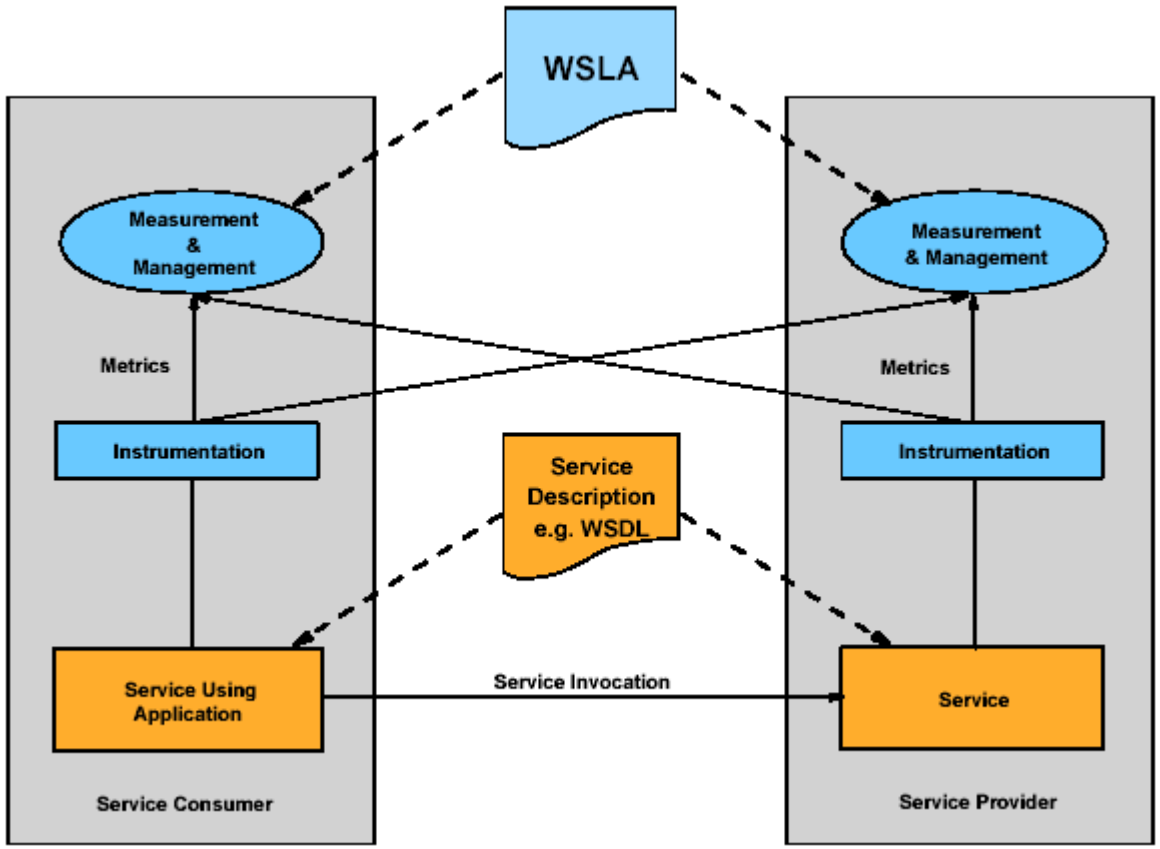
Service Description (1)

- A WSDL document contains seven sections:
  - Types : data type definitions.
  - Message : exchanged message definitions.
  - Operation : operation definitions.
  - Port Type : set of operations reachable on one or more access points (endpoints).

Service Description (2)

- Binding : communication protocol and data format used on a given port type.
- Port : access point defined as a combination of a « binding » and a network address.
- Service : set of related access points.

Web Service Level Agreement



Conclusion on Services

- Assets
  - Adaptation to various usage contexts.
- Advantages
  - Decoupling between usage and realization
  - Possible evolution to cooperative applications
    - Peer-to-peer interactions,
    - Complex multi-peer interactions.
- Limits
  - Web Services still highly influenced by their technical basis
    - Web techniques,
    - XML techniques.

Bibliography on Services

- OASIS, « Reference Model for Service Oriented Architecture 1.0 ». soa-rm-cs, August 2006.
- W3C, « Web Services Architecture». wsa, February 2004.

Contents

- Basic principles
- Procedural approach
- Object-oriented approach
- Component-based approach
- Service-oriented approach
- Conclusion

Standards and Products (1)

- Great diversity but four main approaches
  - Procedural
  - Component-based
  - Object-oriented
  - Service-oriented

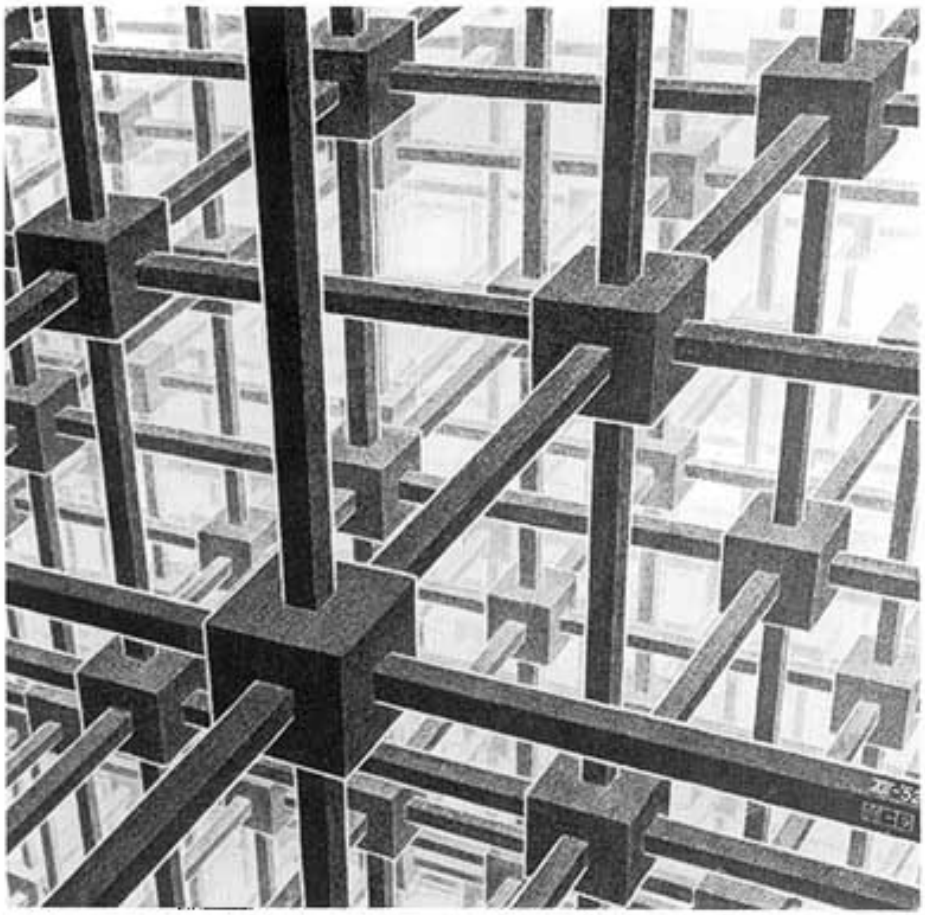
Standards and Products (2)

- Procedural approach
  - TP Monitors,
  - DTP model -> Xa interface,
  - Tuxedo from BEA, CICS from IBM.
- Object-oriented approach
  - Object brokers,
  - CORBA,
  - Orbix from IONA, Visibroker from Borland.

Standards and Products (3)

- Component-based approach
  - Application servers,
  - EJB,
  - WebLogic from BEA, WebSphere from IBM.
- Service-oriented approach
  - SOA : Service Oriented Architecture,
  - Web Services,
  - .Net from Microsoft.

Techniques (1)



- Classification
  - Communication
  - Coordination
  - Conversion
  - Facilitation

Techniques (2)

- Communication functions
  - Control flow with direct binding,
  - Information flow with indirect binding.
- Coordination functions
  - Transaction strategy,
  - Workflow strategy.

Techniques (3)

- Conversion functions
  - Generic framework,
  - Platform abstraction

  … to « modelware ».
- Facilitation functions
  - Quality of Service management

  … to « autonomic » systems.

Thanks for your attention !

Any questions?