中间件技术
Middleware technology

第三章　ODP和Corba介绍

赖永炫　博士/教授
厦门大学　软件工程系

开放分布式处理参考模型

Referenced Model of Open Distributed Processing

概念

开放分布式处理参考模型(Referenced Model of Open Distributed Processing, RM-ODP)是一种标准，规定了使用于开放式分布处理领域内的其他标准确必须遵循的参考模型
创建一个面向应用的参考模型来对付分布的应用，它能使应用之间实现互通和互操作

ODP is ISO's standardisation effort for distributed processing. ODP is a set of standards produced by ISO/IEC and ITU-T. ODP is the ISO/IEC set of standards 10746, and ITU-T's X.900.

International Organization for Standardization国际标准化组织
International Electrotechnical Commission国际电工委员会
International Telecommunications Union 国际电信同盟

ODP标准的组成

1、观点 Viwepoint

2、透明性 Transparencies

观点

观点把对于一个系统的说明分成若干个不同的侧面。每个观点对同一个分布式系统的某个不同侧面进行描述。

ODP viewpoints provide a framework for specifying ODP systems.

The viewpoints of ODP-RM have two main uses: firstly, as a framework to specify other ODP component standards; and secondly, ODP viewpoints are a useful framework for system development, whether the system is an ODP system or not.

观点

企业观点 Enterprise viewpoint:

信息观点 Information viewpoint:

计算观点 Computational viewpoint:

工程观点 Engineering viewpoint:

技术观点 Technology viewpoint:

Enterprise viewpoint: focuses on the purpose, scope, and policies of a system. It provides the context and overall environment within which the system will be built, and therefore constraints, and obligations that must apply in all other viewpoints.

Purpose, scope, roles, policies, activities, environment contract, communities, federation

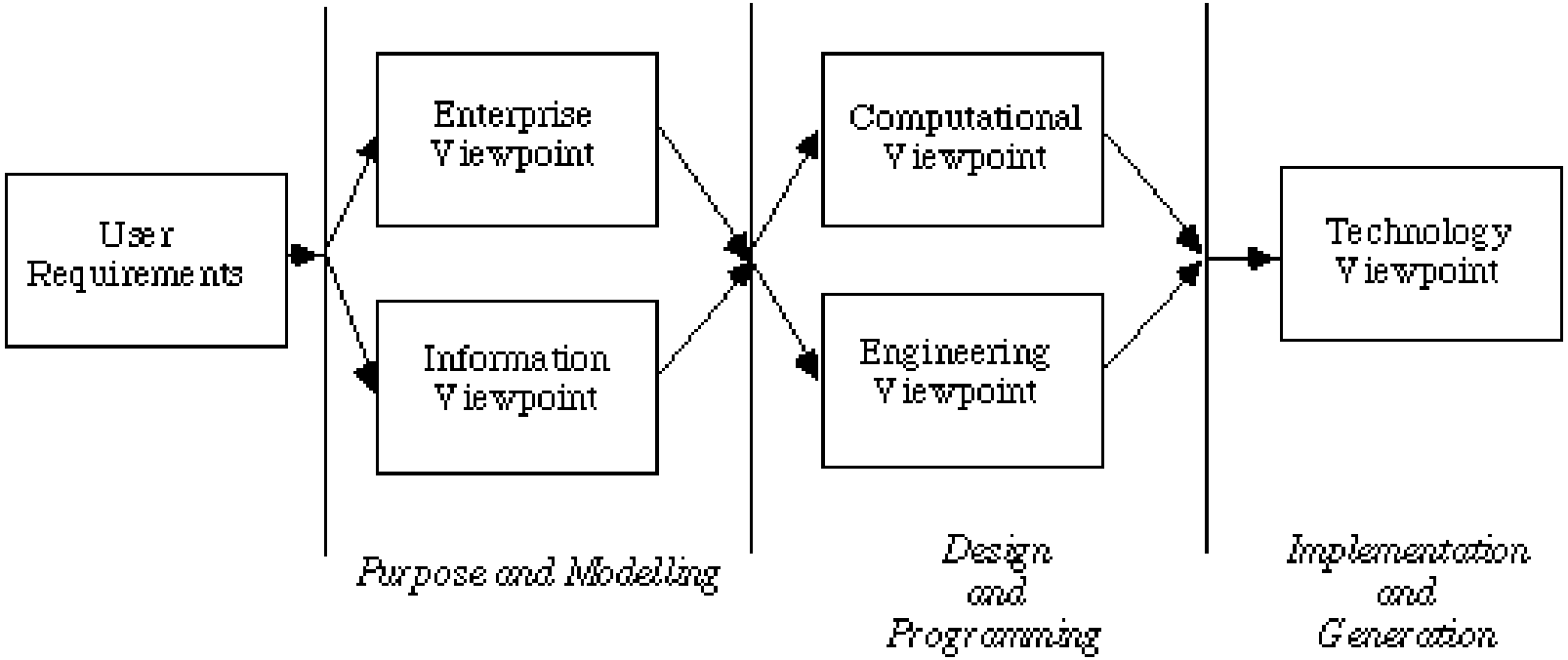
Information viewpoint: focuses on the information and associated processing of a system. The information viewpoint uses schemata to specify the way that information may be transformed.

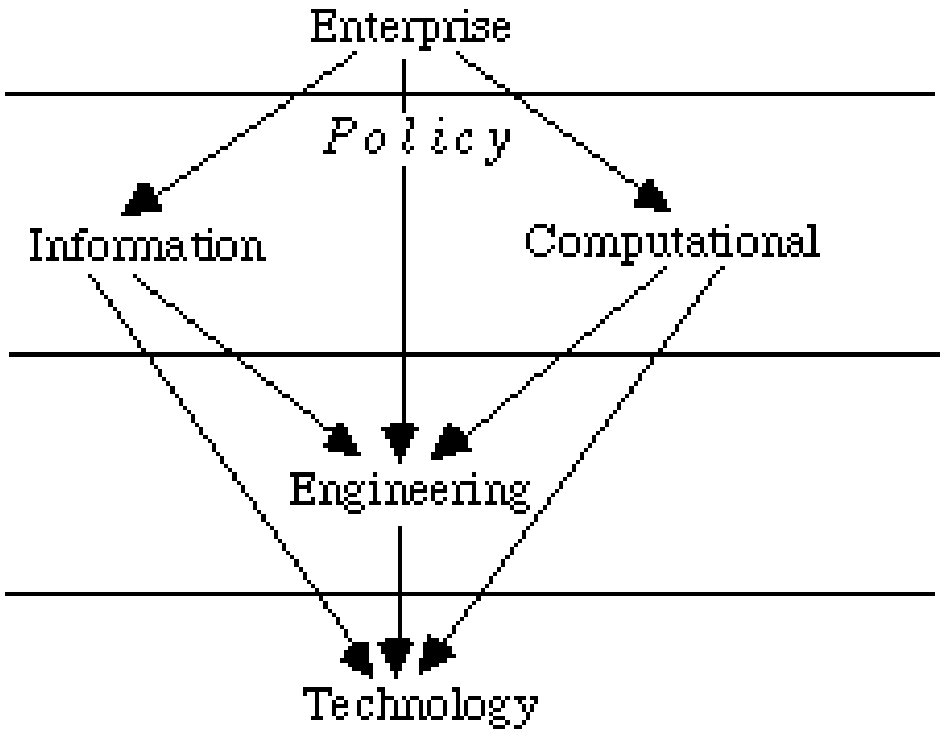
Computational viewpoint: provides functional decomposition of a system into objects that interact at interfaces. These objects will provide natural lines along which a system may be partitioned for distribution.

Engineering viewpoint: focuses on the deployment aspects of a system. In contrast to the computational viewpoint which merely implicitly enables distribution, distribution is explicit in the engineering viewpoint.

Technology viewpoint: details specific technologies, both hardware and software which will be used to implement a system. The technology viewpoint fills in specifics for particular implementations of a system.

观点的联系


User Requirements
Enterprise Viewpoint
Information Viewpoint
Computational Viewpoint
Engineering Viewpoint
Technology Viewpoint
Purpose and Modelling
Design and Programming
Implementation and Generation



Requirements Analysis

Functional Specification

Design

Implementation

透明性 Transparencies

透明性屏蔽了由系统的分布所带来的复杂性

The basis of ODP's ease of use is that programmers and end users should not need to be concerned with the nature and means of distribution.

In other words, programming and use of a distributed application appears exactly the same as if the application were not distributed at all.

透明性

访问透明性 access transparency;
位置透明性 location transparency;
迁移透明性 migration transparency;
失败透明性 failure transparency;
重定位透明性 relocation transparency;
复制透明性 replication transparency;
持久透明性 persistence transparency;
事务处理透明性 transaction transparency.

透明性 Transparencies

From the end users point of view, transparencies determine how they see the system, or more precisely, what they don't see of the system.

It provides the same kind of masking users from technical details that do not concern users to access and utilize a service.

You should note that not all ODP transparencies might be relevant to a particular application.

ODP功能 Functions

While ODP transparencies provide insight into what a distributed system looks like, ODP functions provide some building blocks to assemble ODP systems.

功能组成

The ODP functions are categorized into four groups:

管理功能 management functions;

协作功能 coordination functions;

仓库功能 repository functions;

安全功能 security functions.

Management functions

node management function;

capsule management function;

cluster management function;

object management function.

Coordination Functions

event notification function;
checkpoint and recovery function;
deactivation and reactivation function;
group function;
replication function;
migration function;
engineering interface reference tracking function;
transaction function;
ACID transaction function.

Repository Functions

storage function;

information organization function;

relocation function;

type repository function;

trading function.

Security Functions

access control function;
security audit function;
authentication function;
integrity function;
confidentiality function;
non-repudiation function;
key management function.

ODP与其他规范的关系

ODP is independent of any particular network, and may be used with OSI, TCP/IP, NetBIOS, and other networks. There is therefore no relationship between ODP and other network standards.

ODP与其他规范的关系

不仅是一个一般标准，还是一个标准的标准(meta-standard)，即规定了使用开放分布式处理领域内的其他标准必须遵循的参考模型

ODP is not in competition with other standards. In fact ODP is very much the big picture of distributed processing, and CORBA and DCOM fit into the ODP framework very well.

ODP 小结

OPEN DISTRIBUTED PROCESSING is ISO's standardisation effort for distributed processing.

ODP标准的组成

1、观点 Viwepoint
2、透明性 Transparencies

You should take a look at ODP if you are a systems architect, programmer or a manager who wants an understanding of the structure of modern systems development, and useful frameworks for development processes.

CORBA 介绍

https://www.corba.org/orb_basics.htm

定义

◆CORBA（Common Object Request Broker Architecture，公共对象请求代理体系结构）是由OMG（对象管理组织，Object Management Group）提出的应用软件体系结构和对象技术规范。

◆其核心是一套标准的语言、接口和协议，以支持异构分布应用程序间的互操作性及独立于平台和编程语言的对象重用。

◆CORBA使用面向对象模型实现分布式系统中的透明服务请求。

对象管理小组OMG

对象管理组（Object Management Group, OMG）是一个非赢利性的协会组织，组建于1989年，由一些的计算机公司发起，目前成员已超过800个， 遍及计算机制造商、软件公司、通信公司和最终用户。

对象管理小组OMG

为使该组织所采纳的技术具有开放性，OMG所采用的方法是，针对某一领域发出RFP（Request For Proposal），然后以各方提交的建议为基础，经过一系列的讨论和协商，产生最终的规范。

CORBA规范主要基于以下几个公司所提交的建议：DEC、HyperDesk、HP、SunSoft、NCR和Object Design。

其目的是，为面向对象的应用提供一个公共框架，如果符合这一框架，就可以在多种硬件平台和操作系统上建立一个异质的分布式应用环境。

对象管理体系结构

由OMG制定的最关键的规范——对象管理结构（Object Management Architecture, OMA）和它的核心（也就是CORBA规范），提供了一个完整的体系结构。

这个结构以足够的灵活性、丰富的形式适用了各类分布式系统。

对象管理体系结构

OMA描述了面向对象技术在分布式处理中的运用。它包括两部分：对象模型（Object Model）和参考模型（Reference Model）。

对象模型：定义如何描述分布式异质环境中的对象；

参考模型：描述对象之间的交互。

OMA对象模型

在OMA对象模型中，对象是一个被封装的实体，它具有一个不可改变的标识，并能给客户用户提供一个或多个服务。

```
interface printer
{
        attribute model;
        void print(in string buffer);
};
```

OMA对象模型

对象的访问方式是通过向对象发出请求来完成的。请求信息包括目标对象、所请求的操作、0个或多个实际参数和可选的请求上下文（描述环境信息）。每个对象的实现和位置，对客户都是透明的。

OMA参考模型

在OMA参考模型中，OMG定义了一条为对象所公用的通信总线，即ORB(Object Request Broker)。

同时， OMG又定义了对象进出这一总线的界面。这包括：

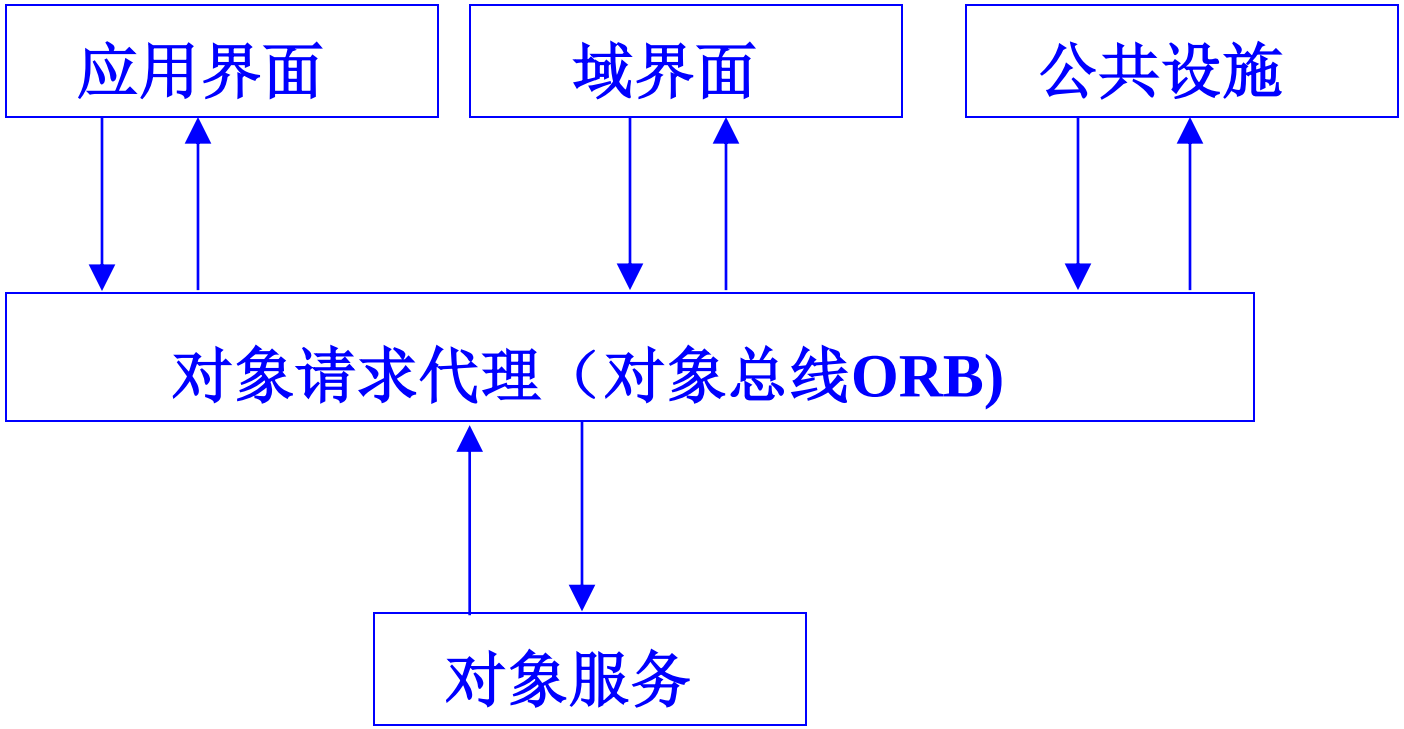
对象服务（Object Services）

公共设施（common facilities）

应用界面（Application Interface）

域界面（Domain Interface）

OMA参考模型



对象总线ORB

ORB(Object Request Broker)是对象总线。通过ORB，一个client可以透明的调用同一台机器上或网络上的一个server对象的方法。

ORB解释该调用并负责查找一个实现该请求的对象，找到后，把参数传给该对象，调用它的方法，最后返回结果。

客户方不需要了解服务对象的位置、通信方式、实现、激活或存储机制。

对象服务

对象服务提供基本服务，与具体的应用领域无关的界面。目前，CORBA支持的这类服务有：

命名服务（Naming Service）：允许通过名字查找对象。

持久性服务（Persistence Service）:提供在各种存储服务器（包括对象数据库、关系数据库和简单文件）上永久性存储对象的统一界面。

对象服务

生存周期服务（Life Cycle Service）：定义了在对象总线上创建、拷贝、移动和删除对象的操作。

事务处理服务（Transaction Service）：提供两阶段提交协议，用于确保ORB上的一些分布式对象协同地完成事务处理。

事件服务（Event Service）：允许对象动态注册或撤消指定的事件。

安全服务（Security Service）：提供一个分布式对象安全的完整框架。

公共设施

与对象服务不同的是，公共设施面向最终用户的应用，它是各种应用可以共享的一系列服务集合。

比如，复合文档的管理工具，数据库存取工具、文件打印工具、电子邮件服务都等属于公共设施。

域界面

针对着某一特殊的应用领域。例如，PDME(工厂数据管理环境) 是OMG发出的最早的这类RFP之一，它是为解决制造领域中的问题而发出的。另外，OMG也已经发出了通信、医药和财务等领域中的这类RFP

应用界面

应用界面针对某一具体应用而产生。

内容

对象管理体系结构（OMA）

CORBA的组成结构

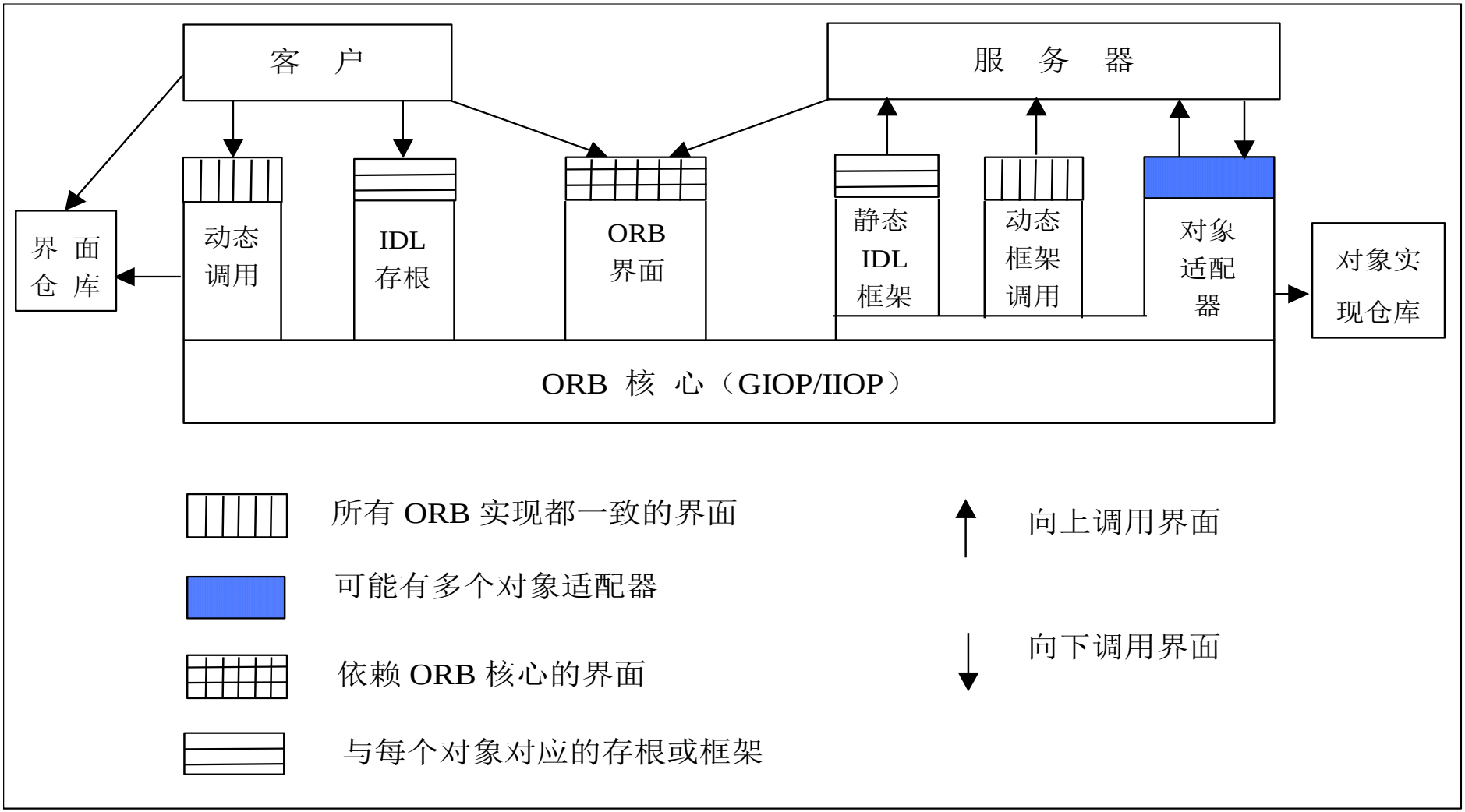
ORB核心、IDL语言和语言映射、存根和框架、动态调用、对象适配器、界面仓库和仓库实现、ORB之间的互操作

公共对象请求代理体系结构（CORBA）

CORBA规范详细说明了OMA中ORB组件的特性和界面。最新的CORBA规范主要包含以下内容：

- ORB 核心（ORB CORE）
- OMG界面定义语言
- 界面仓库和实现仓库
- 语言映射
- 存根和框架
- 动态调用
- 对象适配器（Object Adapter）
- ORB之间的互操作

CORBA


客 户
服 务 器
界 面
仓 库
动态
调用
IDL
存根
ORB
界面
静态
IDL
框架
动态
框架
调用
对象
适配
器
对象实
现仓库
ORB 核 心（GIOP/IIOP）
所有 ORB 实现都一致的界面
可能有多个对象适配器
依赖 ORB 核心的界面
与每个对象对应的存根或框架
向上调用界面
向下调用界面

ORB 核心

功能：把客户发出的请求传递给目标对象，并把目标对象的执行结果返回给发出请求的客户。其重要特征是：提供了客户和目标对象之间的交互透明性。这主要包括：

1，对象位置：客户不必知道目标对象的物理位置。它可能与客户一起驻留在同一个进程中或同一机器的不同进程中，也有可能驻留在网络上的远程机器中。

ORB 核心

2，对象实现：客户不必知道有关对象实现的具体细节。例如，设计对象所用的编程语言、对象所在节点的操作系统和硬件平台等。

3、对象的执行状态：当客户向目标对象发送请求时，它不必知道当时目标对象是否处于活动状态（即是否处于正在运行的进程中）。此时，如果目标对象不是活动的，在把请求传给它之际，ORB会透明地将它激活。

ORB 核心

4、对象通信机制：客户不必知道ORB所用的下层通信机制，如，TCP/IP、管道、共享内存、本地方法调用等。

5、数据表示：客户不必知道本地主机和远程主机对数据表示方式，如高位字节在前还是在后等，是否有所不同。

IDL语言和语言映射

在客户向目标对象发送请求之前，它必须知道目标对象所能支持的服务。对象是通过界面定义来说明它所能提供的服务。CORBA对象的界面是利用OMG IDL来定义。

Interface definition language

IDL语言和语言映射

OMG IDL 的语法与C++类似（包括C++的预处理语句），它另外增加了一些支持分布式处理的关键字（in、out和inout等）。OMG IDL 不是编程语言，而是一个纯说明性语言，并且与具体的宿主语言（主机上的编程语言）无关。这就很自然地将界面与对象实现分离，使得可以用不同的语言来实现对象，而它们之间却又可以进行互操作。

IDL语言和语言映射

不能用OMG IDL 直接去实现分布式应用，需要把IDL的特性映射为具体语言的实现，这就是语言映射的任务。到目前为止，OMG已为C、C++、SmallTalk、Ada95、Cobol和Java制定了语言映射标准。

存根(stub)和框架(skeleton)

除了把IDL的特性映射为具体语言外，OMG IDL编译器还根据界面定义来产生客户方的存根和服务方的框架。

存根的作用是代表客户创建并发出请求；框架的作用则是把请求交给CORBA对象实现。

存根(stub)和框架(skeleton)

具体地说，存根使得客户能够不关心ORB的存在，而只要把请求交给存根，存根则负责对请求参数进行封装和发送，以及对返回结果进行接收和解封装。

框架在请求的接收端提供与存根类似的服务，它将请求参数解封装，识别客户所请求的服务，调用对象实现，并把执行结果封装，然后返回给客户方。

存根和框架

由于存根和框架都是从用户的界面定义编译而来，所以它们都与具体的界面有关，并且，在请求真实发生之前，存根和框架早以分别被直接连接到客户程序和对象实现中去。为此，通过存根和框架的调用被通称为静态调用。

动态调用

CORBA还支持两种用于动态调用的界面：

动态调用界面（DII）——支持客户方的动态请求调用。

动态框架界面（DSI）——支持服务方的动态对象调用。

动态调用

利用DII，客户方应用可以在运行时动态地向任何对象发出请求，而不象静态调用那样，必须在编译时就知道特定的目标对象的界面信息。使用DII时，用户必须手工构造请求信息，包括相应的操作及有关参数等。类似地，DSI允许用户在没有静态框架信息的条件下来获得对象实现。

接口仓库和实现仓库

ORB提供了两个用于存储有关对象信息的服务：界面仓库和实现仓库。

接口仓库

界面仓库存储各个界面信息的模块，如用IDL编写的界面定义、常量、类型等。它本身作为一个对象而存在。应用程序可以象调用其它CORBA对象所提供的操作一样，来调用界面仓库的操作。界面仓库允许应用程序在运行时访问OMG IDL类型系统。例如，当应用程序在运行时遇到一个不知道其类型的对象时，可以通过界面仓库的操作来遍历系统中的所有界面信息。由此可见，界面仓库的引入很好地支持了CORBA的动态调用。

接口仓库

IDL:CCS:1.0

IDL:CCS/Temptype:1.0

IDL:CCS/Thermometer:1.0

IDL:CCS/Thermometer/temperature:1.0

IDL:CCS/Thermostat:1.0

IDL:CCS/Thermostat/set_temp:1.0

实现仓库

实现仓库所完成的功能与界面仓库类似，只不过它存储的是对象实现的信息。当需要激活某一对象类型的实例时，ORB需要访问这些信息。

对象适配器

对象适配器是联系对象实现与ORB本身的纽带。另外，它的引入还大大减轻了ORB的任务，从而简化了ORB的设计。具体地说，对象适配器主要完成以下工作：

1、对象登记——利用对象适配器所提供的操作，可以将CORBA的实现仓库中具有编程语言形式的实体（servant，伺服器）登记为CORBA的对象实现。

对象适配器

2、对象引用(OR,Object Reference)的产生——对象适配器为CORBA对象生成对象引用。客户应用程序通过对象引用来访问对象实例。

3、服务器进程的激活——如果客户发出请求时，目标对象所在的服务器还未运行，则对象适配器自动激活该服务器。

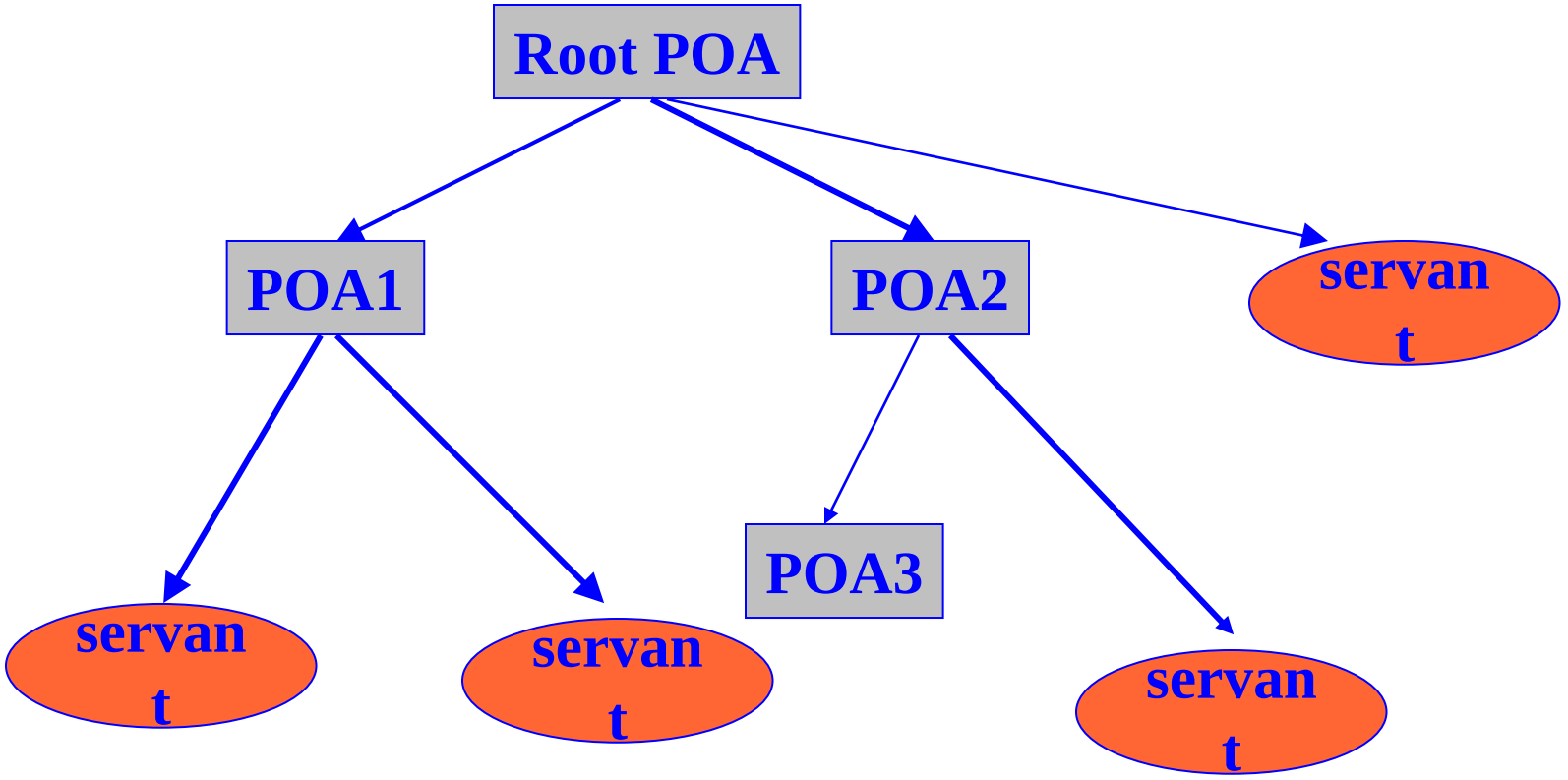
对象适配器

4、对象的激活——如有必要，自动激活目标对象。

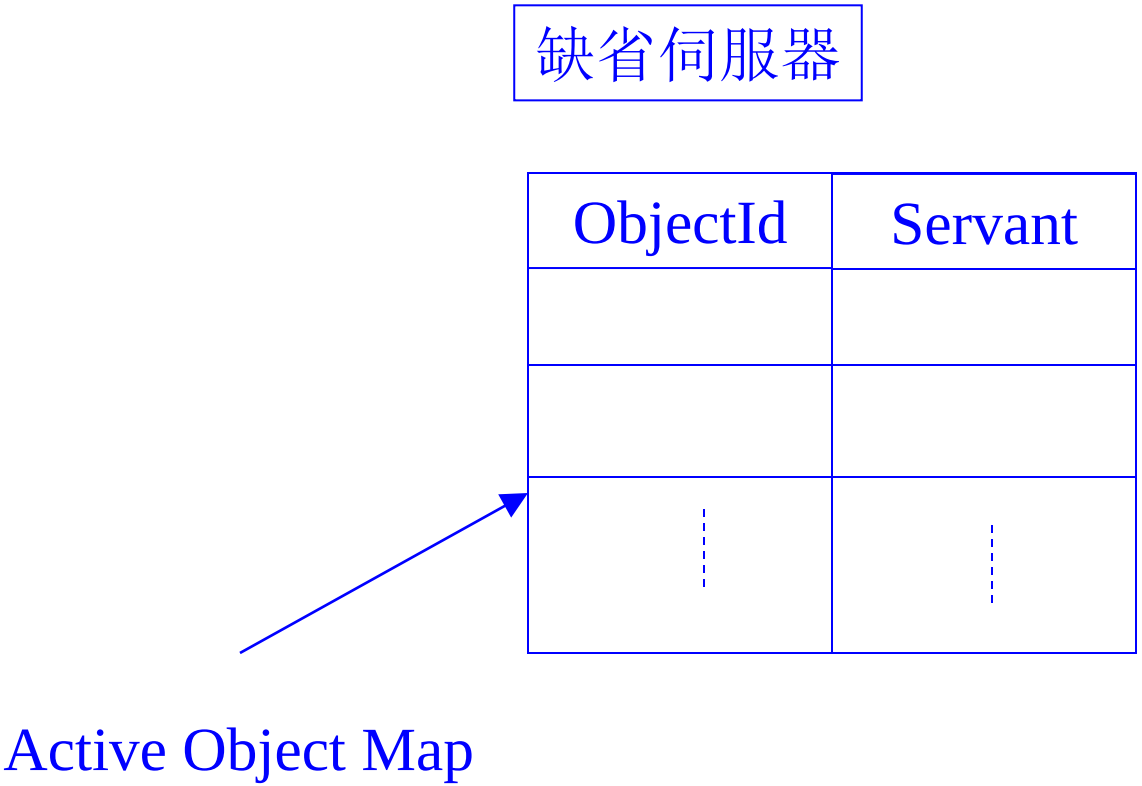
5、对象的撤消——在预先规定的时间片内，如果一直没有发向某个目标对象的请求，则对象适配器撤消这一对象，以节省系统资源。

6、对象向上调用——对象适配器把请求分配给已登记了的对象。

对象适配器

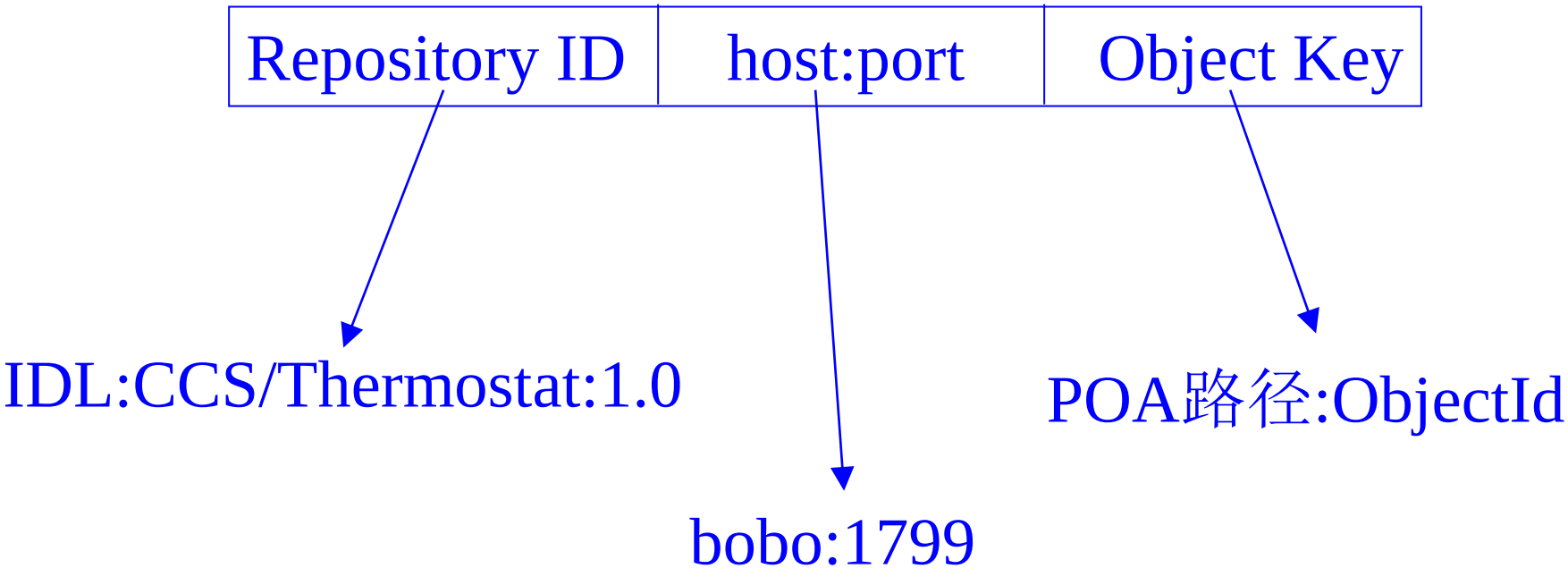


对象适配器

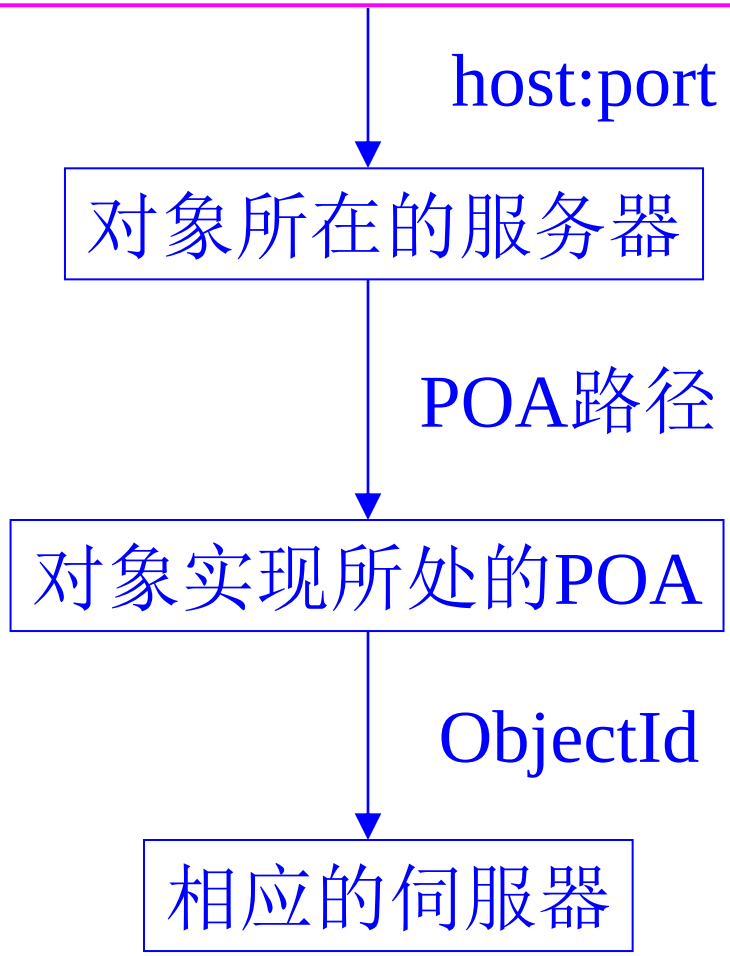


对象引用格式


对象引用
ObjectId
Servant


Repository ID
host:port
Object Key
IDL:CCS/Thermostat:1.0
bobo:1799
POA路径:ObjectId

对象引用到伺服器



对象引用的获取

对象引用是客户机获取目标对象唯一途径。客户机获取引用的方式：

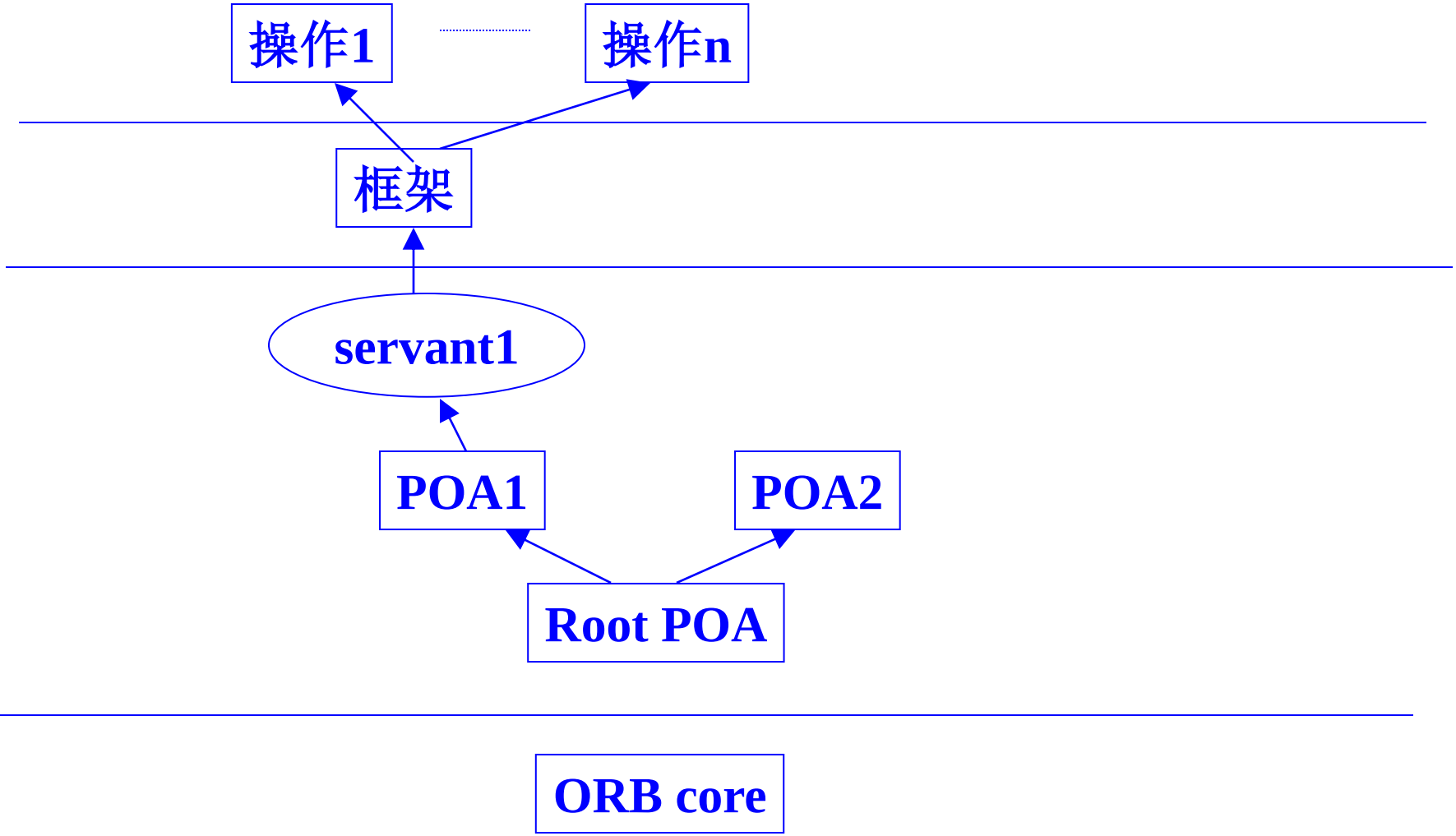
1、通过调用公共操作list_initial_services、resolve_initial_services来获取标准服务应用，如“RootPOA”、“NameService”、“TradingService”等。

2、以某些已知的服务程序公告一个引用，如名字服务

3、通过将对象引用转换成一个字符串和将它写入一个文件，来公布一个对象引用。返回一个引用作为一个操作的结果（就像返回一个值，或返回一个inout或out参数）。

4、通过其他可以外传的方式来传送一个对象引用，比如，用电子邮件发送或者在Web网页上公布。

对象适配器之请求处理流程



ORB 之间的互操作

在发布CORBA2.0之前，ORB产品的最大缺点是：不同厂商所提供的ORB产品之间并不能互操作。为了达到异构ORB系统之间互操作的目的，CORBA2.0规范中定义了标准通信协议GIOP（General inter-ORB Protocol）。

ORB 之间的互操作

GIOP协议由3个部分组成：

1、公共数据表示（Common Data Representation，简称CDR）；

2、GIOP消息格式： 它定义了用于ORB间对象请求、对象定位和信道管理的7种消息的格式和语义

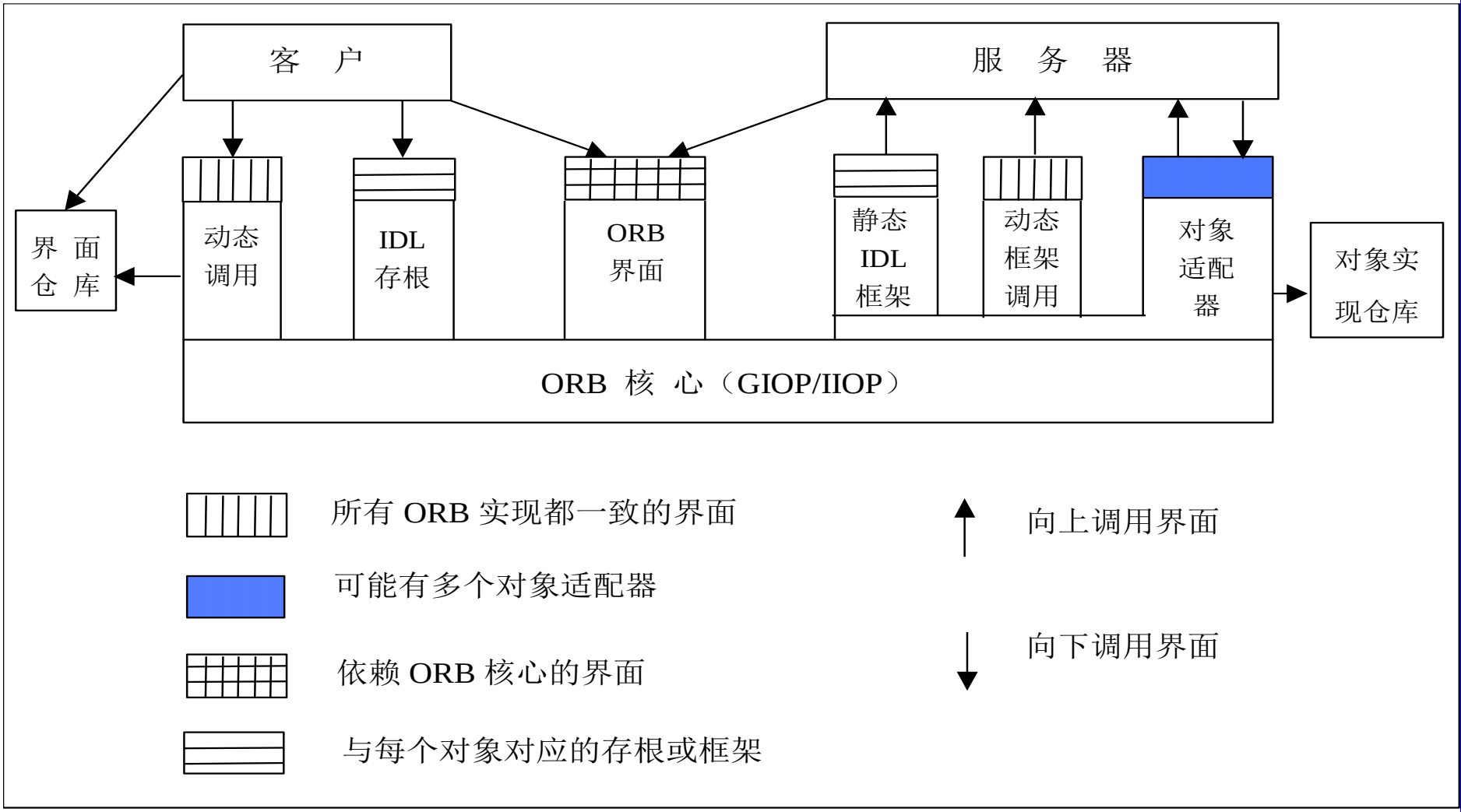
ORB 之间的互操作

3、传输层假设，GIOP协议可运行于多种传输层协议之上，只要传输层协议是面向连接的、可靠的，所传递的数据可以为任意长度的字节流，提供错序通知功能，连接的发起方式可以映射到TCP/IP这样的一般连接模型。

ORB 之间的互操作

GIOP协议只是一种抽象协议，独立于任何特定的网络协议，在实现时必须映射到具体的传输层协议或者特定的传输机制之上。GIOP协议到TCP/IP协议的映射又称为IIOP（Internet Inter-ORB Protocol）协议。

CORBA


客 户
服 务 器
界 面
仓 库
动态
调用
IDL
存根
ORB
界面
静态
IDL
框架
动态
框架
调用
对象
适配
器
对象实
现仓库
ORB 核 心（GIOP/IIOP）
所有 ORB 实现都一致的界面
可能有多个对象适配器
依赖 ORB 核心的界面
与每个对象对应的存根或框架
向上调用界面
向下调用界面

CORBA的发展

CORBA version 1.0 (developed in 1991) –

ORB used own proprietary communication protocol. CORBA对象模型、IDL、用于动态请求管理和动态调用的API集合和界面仓库。

CORBA1.1

引入对象适配器概念，并提供了BOA。

CORBA version 2.0 (developed in 1995) –

Protocols used:

Standard General Inter-ORB (GIOP)

Internet Inter ORB protocol (IIOP)

增加了跨平台ORB的互操作规范

CORBA version 2.1 –

BOA (Basic Object Adapter) was developed

CORBA的发展

CORBA version 2.2 –

the Portable Object Adapter (POA)，解决了不同CORBA平台上应用的可移植性

CORBA2.3

增加了用于嵌入式设备的miniCORBA规范。

CORBA2.4

增加了服务于实时应用的rtCORBA规范。

CORBA version 3.0 – New features added

CORBA Messaging

Objects by Value

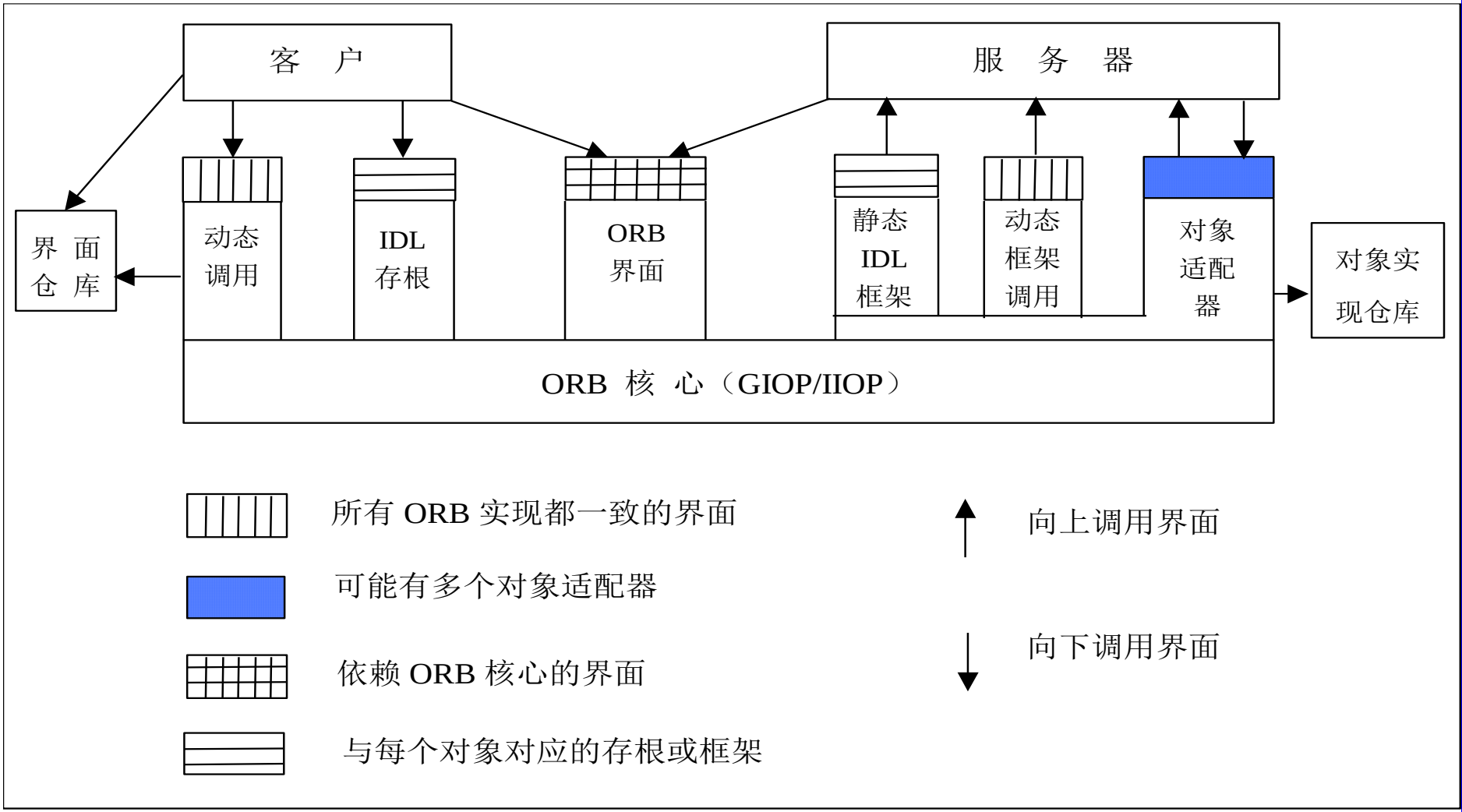
CORBA产品

比较著名的有IONA的Orbix、ExpertSoft的Power CORBA以及Inprise的Visibroker。同时，还有一些优秀的成果可供研究，如Mico, Orbacus, TAO等。

小结

- CORBA是应用软件体系结构和对象技术规范
- 其核心是一套标准的语言、接口和协议，以支持异构分布应用程序间的互操作性及独立于平台和编程语言的对象重用
  - ✓ ORB 核心（ORB CORE）
  - ✓ OMG界面定义语言
  - ✓ 界面仓库和实现仓库
  - ✓ 语言映射
  - ✓ 存根和框架
  - ✓ 动态调用
  - ✓ 对象适配器（Object Adapter）
  - ✓ ORB之间的互操作

小结：CORBA规范的主要内容



不足和缺陷

尽管有多家供应商提供CORBA产品，但是仍找不到能够单独为异种网络中的所有环境提供实现的供应商。不同的CORBA实现之间会出现缺乏互操作性的现象，从而造成一些问题；而且，由于供应商常常会自行定义扩展，而CORBA又缺乏针对多线程环境的规范，对于像C或C++这样的语言，源码兼容性并未完全实现。

CORBA过于复杂，要熟悉CORBA，并进行相应的设计和编程，需要许多个月来掌握，而要达到专家水平，则需要好几年。

谢谢