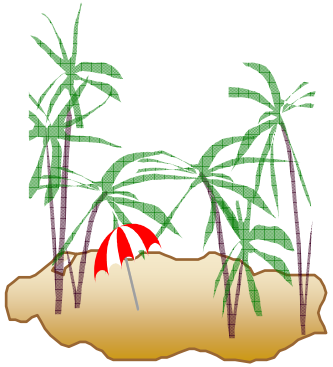
Middleware technology
中间件技术
（软件技术本质）

- Software School of Hunan university -

Jin-Min Yang
（杨金民）

2013/02



Course Information

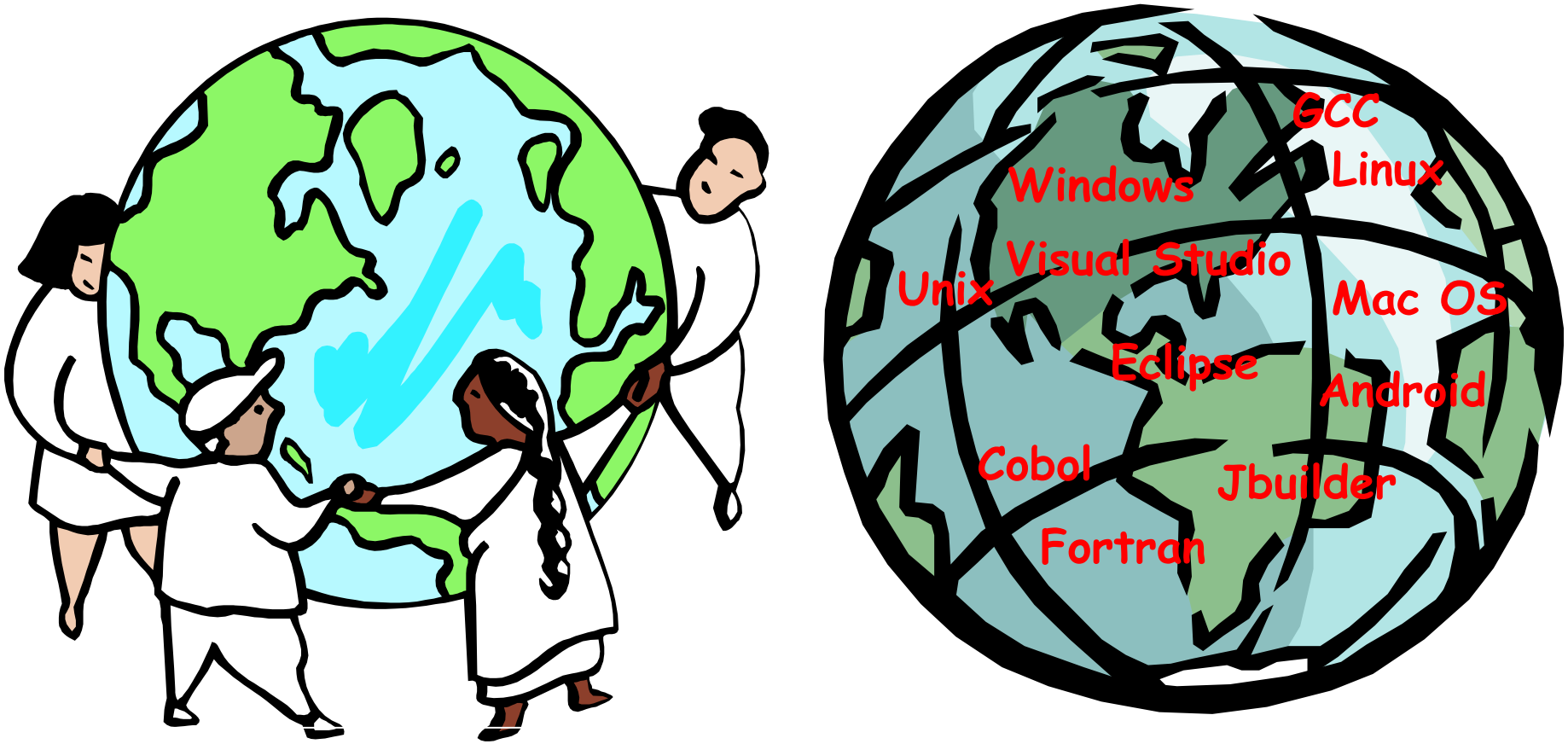
- 上课老师：杨金民 教授
- 办 公 室：湖南大学软件大楼507
- 电　　话：13975896967，0731-8711167
- Email：rj_jmyang@hnu.edu.cn
- QQ：　909485030;
- 网站：
  http://ss.hnu.cn/newweb/teachers/zhuanzhijiaoshi/yangjinmin/index.htm

Chapter 1 Introduction to Middleware technology (essence of software technology)

章节提纲

- 该课程要探讨的问题；
- 软件技术发展的现状和特征；
- 软件技术发展的推动力；
- 软件技术发展趋势；
- 软件危机解决的出路—软件互操作；
- 互操作的基本概念和框架；
- 互操作和中间件技术；
- 中间件的特性；
- 中间件技术的发展历史；

软件技术发展的背景



人类在各地分别起源，**各自有不同的符号和方式**。全球化的到临，要求形成地球村，彼此沟通交流。面临着**语言**和**行为方式**的**差异问题**。须要在此两个方面达成**共识，解决沟通交流问题**。软件的发展也是如此。

软件技术发展的特征

- 软件的出现还只有50多年的历史，是一种新兴的产业。在没有条条框框、标准协议的情况下，各路软件天才各显神通，呈现出百花齐放的局面、没有考虑整合、兼容、互操作问题。
  - 操作系统五花八门：Unix, DOS, Windows, Machitosh, Linux；
  - 程序语言形形色色：Fortran, Cobol, Pascal, Bacic, C, C++, Java, Object Pascal, C#, Script language.
  - 函数调用规范各持己见：_cdecl, _stdcall, _fastcall , thiscall， _pascal, _fortran, _syscall.
  - 编程模式后浪催前浪：Procedure-Oriented, Object-oriented, Component-oriented, Service-oriented.
  - 开发工具多种多样：Microsoft Studio, Eclipse, Borland, Apple.

软件的发展是旧貌换新颜

当一种软件技术**打下一遍江山**后，它又**面临着新问题**，被新的软件技术所取代。其发展是**一浪接一浪，后浪催前浪**：

- 新技术并不意味要推翻和抛弃已有的应用系统，因为**用户习惯，成本，系统可靠性**等等原因。
- 已有的应用系统继续发挥着不可替代的作用，是宝贵的资源。
- 新的应用系统不是要**抛弃旧的应用系统**，而是要**集成旧系统**，达到**旧貌换新颜**的功效。
  - 高级程序语言**屏蔽**了操作系统API以及执行模式的差异：C++, Java.
  - 虚拟平台的出现**瓦解**了编程中的机器和操作系统概念；
  - SOA则把软件的重心从计算（功能实现）**转移**到了互操作的语义上；

软件技术发展的四次飞跃

- 面向过程的编程技术Procedure-Oriented；
- 面向对象的编程技术Object-oriented；
- 面向组件的编程技术Component-oriented；
- 面向服务的编程技术Service-oriented；

软件发展面临的挑战、发展的源动力

- 开发周期；
- 开发成本；
- 软件质量；
- 系统可维护性；
- 产品竞争和开发风险；

最小化成本；

- 资源的利用和共享：

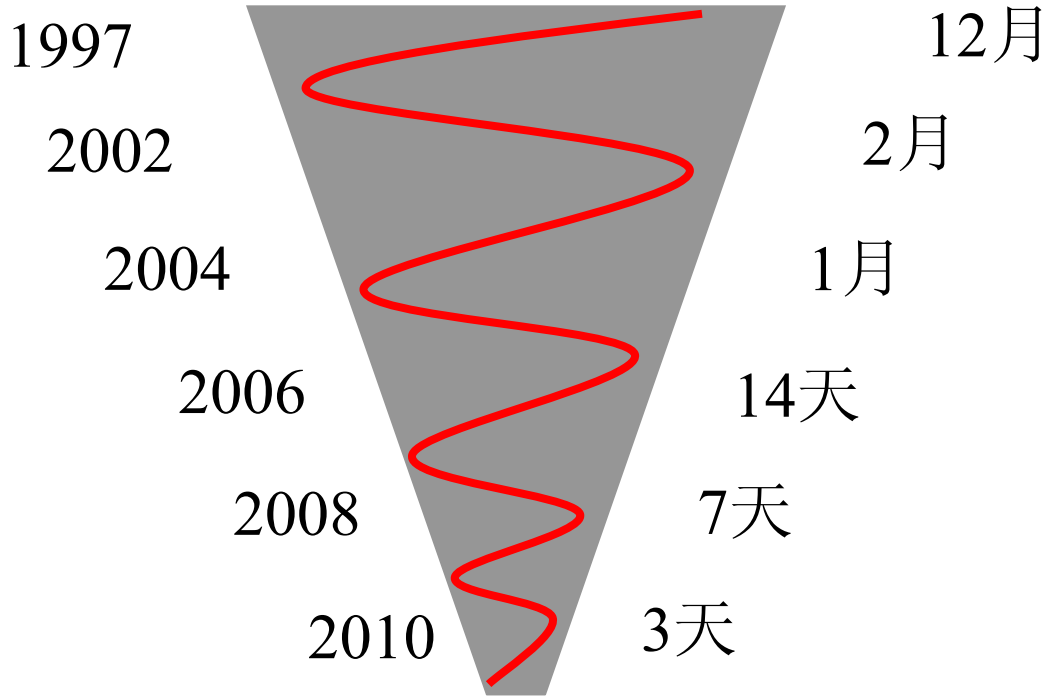
最大化收益；

最强的竞争实力；

软件发展面临的机遇

- Internet的普及又为软件的发展提供巨大机遇：
- 资源的共享使得最大化收益、最小化成本，最快的开发时间成为伸手可及的事情；

快速的开发周期



面临的产品竞争和开发风险

- 第一，软件平台和技术的不断地改朝换代：DOS，Windows，.NET，Java，SOA。
- 第二，随着软件技术愈来愈多，应用也愈来愈多元化。这使得产品的潜在客户群被分散，软件利润趋于下降；
- 第三，信息技术的多元化，产品开发面临难以抉择的状态。面临着**虚拟平台**、**程序语言**和**信息架构**等众多因素的组合变量。

软件危机解决的出路

- **编程的自动化（程序代码的自动生成: 开发工具）**
  - 应用的框架代码由开发工具自动生成，程序员只需要编写功能代码；
- **实现程序的通用性 （实现与操作系统与机器无关）**
  - 操作系统级的源程序；（一个程序要写多遍，多次编译链接）；
  - 与操作系统无关的源程序；（一个程序仅写一遍，基于操作系统和机器要进行多次编译链接）；
  - 与机器无关的源程序；（一个程序只要写一遍，编译链接一遍）；
- **实现软件之间的互操作：（实现以组装模式来开发软件）**
  - 二进制可执行级的软件互操作：同构软件/异构软件之间的互操作；
  - 一个机器中，同进程内互操作；
  - 同一机器中，不同进程间的互操作；
  - 不同机器间的软件互操作；

出路1：编程的自动化（开发工具）

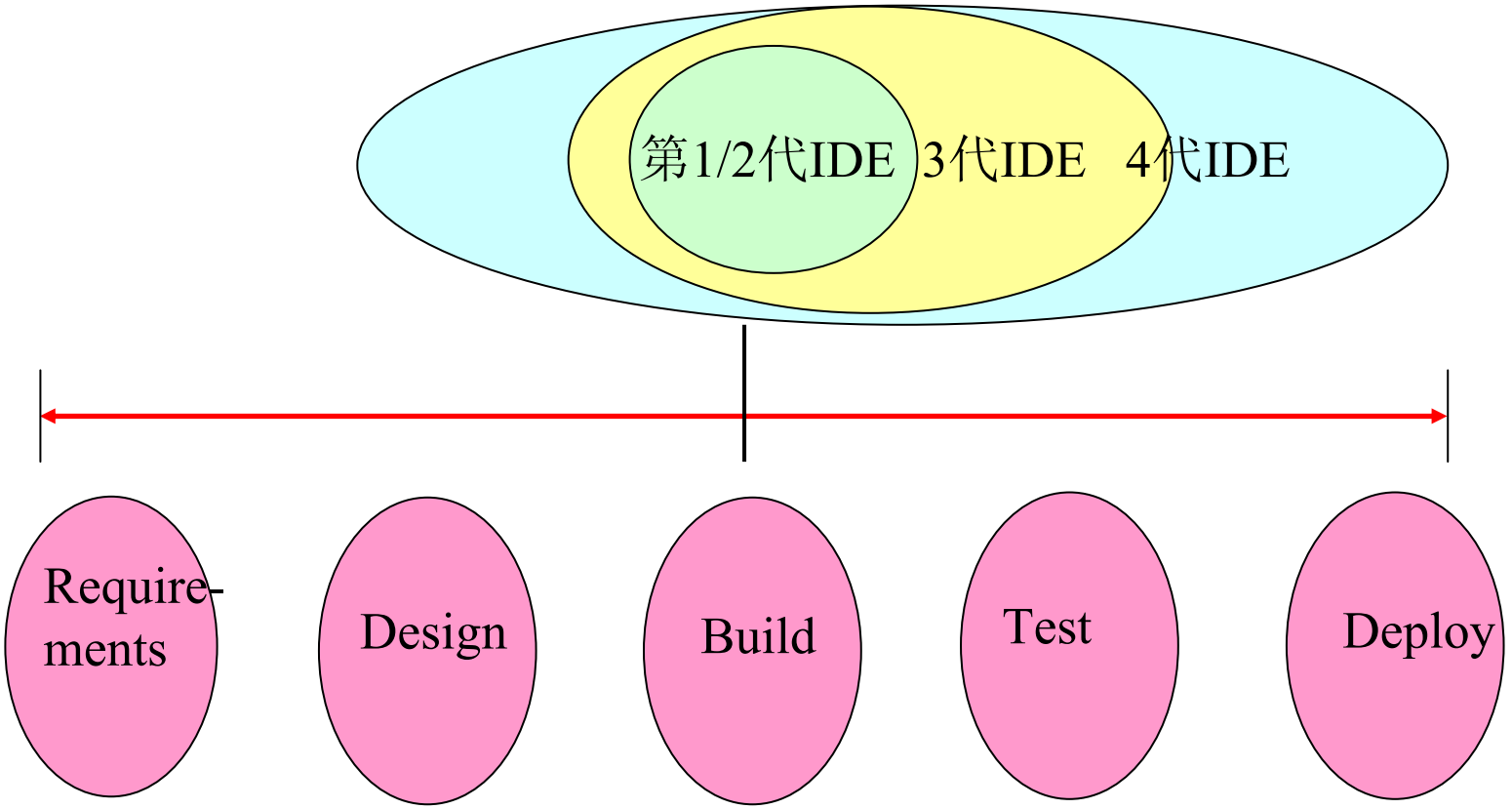
集中开发环境（IDE)从编辑器＋编译器＋连接器， 演变到同时面对专业开发人员，应用开发人员，分析设计人员以及整个开发团队。

集中开发环境仍然是开发周期的核心。提供Framework，以及其他的工具（设计、编程、测试、部署）和Plug-In等。

开发工具提供：

- 基于Framework的代码自动生成；
- 支持库（.h文件，.cpp文件，*.lib(dll) 文件）；
- 编辑、编译、链接；

开发工具的演进

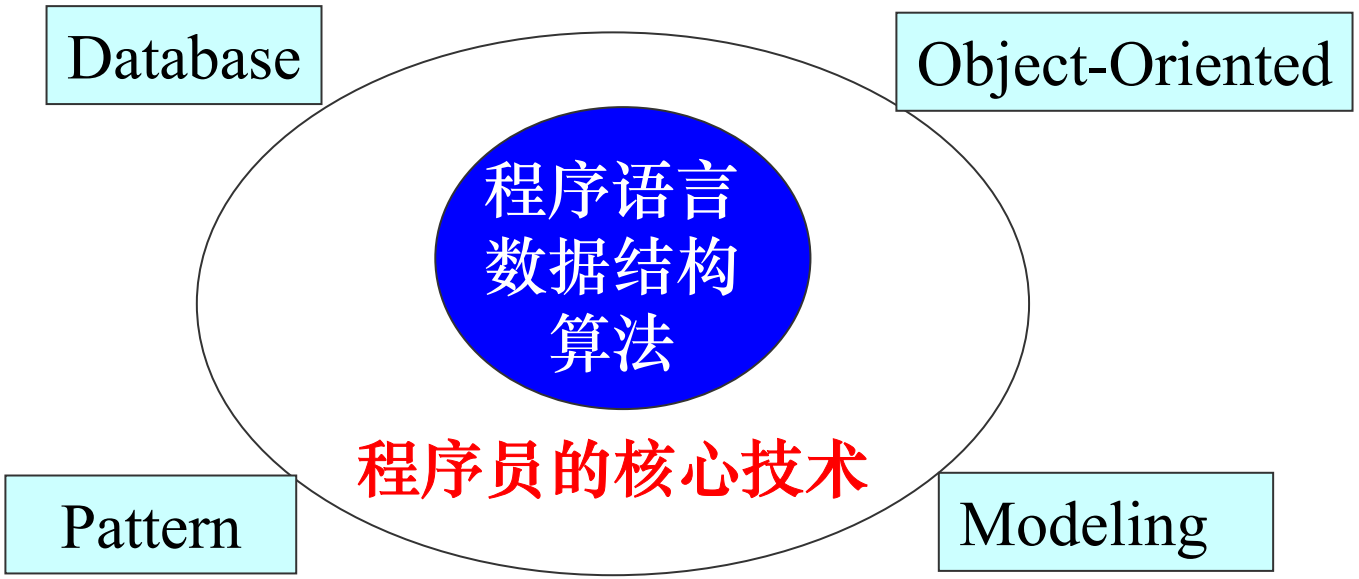

第1/2代IDE
3代IDE
4代IDE
Require-ments
Design
Build
Test
Deploy
Business Manager
Architect
Developer
Tester
Operations

开发工具的演进

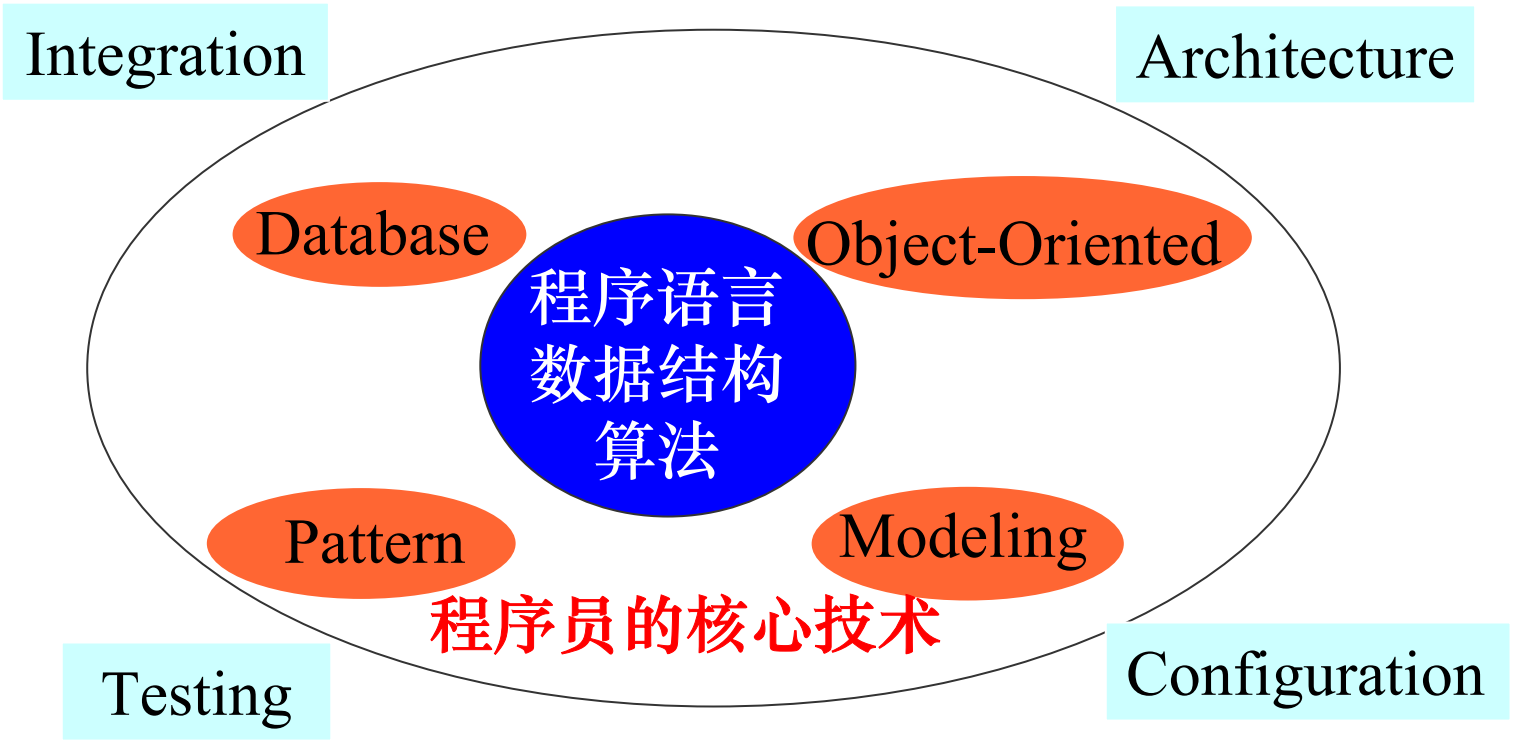
高级语言	第一代IDE Editor + Compiler + Linker	1984	专业工程师
语言/IDE	第二代IDE DOS模式下的IDE	1985 Borland Turbo系列	专业工程师 应用工程师
IDE/Framework MFC/OWL)	第三代IDE GUI模式下的IDE	1993 Borland系列 Microsoft系列	专业工程师 应用工程师 开发团队
语言 /IDE/Framework Java/.NET Modeling/软件工程	第四代IDE IL IDE	2003 Java IDE .NET IDE 2006 Eclipse	专业工程师 应用工程师 开发团队 分析/设计人员

软件开发的知识面

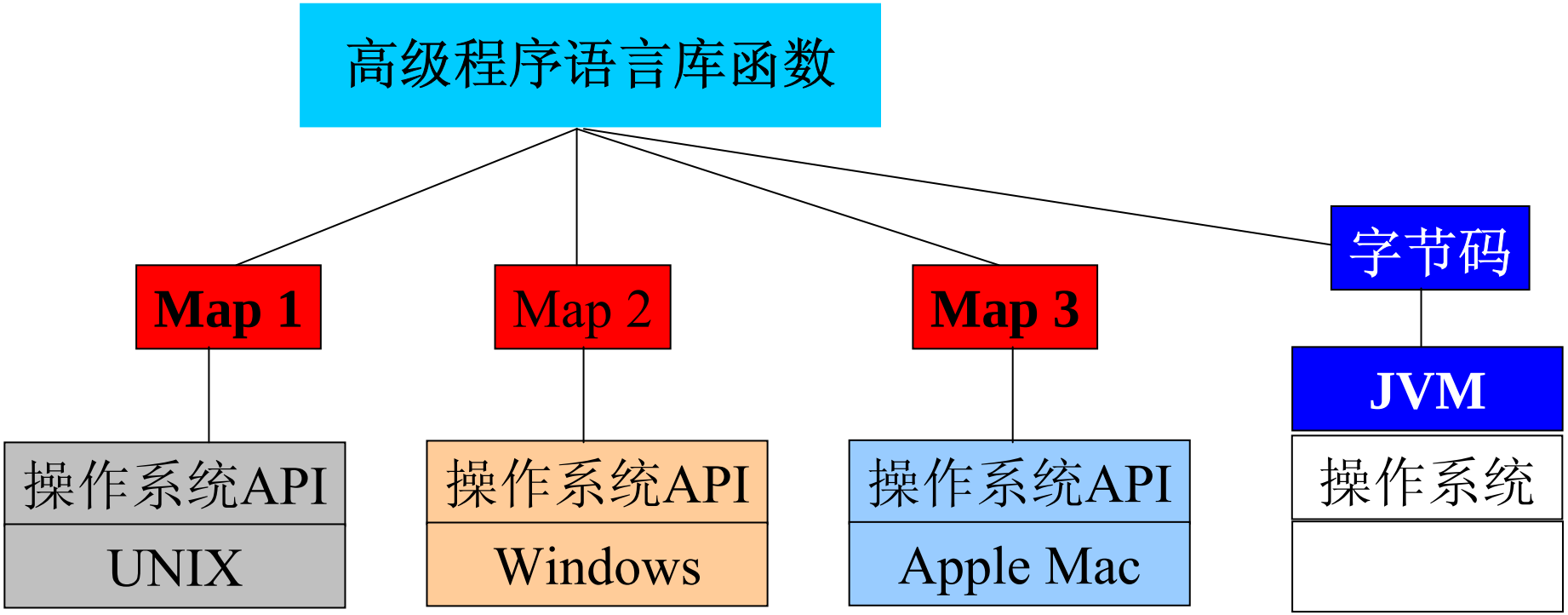
单一程序语言



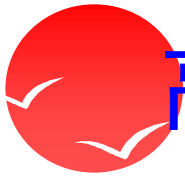
软件开发技能门槛的提升



出路2：实现源程序的通用性



这样，就使得用高级程序语言编写的应用程序与操作系统及机器无关，**实现了源程序的通用性**，一个应用程序只需要编程一遍，便能满足在任何一个机器和操作系统上运行的需要。



高级程序语言到操作系统API的映射例子

- C语言中的库函数：

```
malloc(size_t size);  //分配内存
```

- 在Windows中的Map实现：

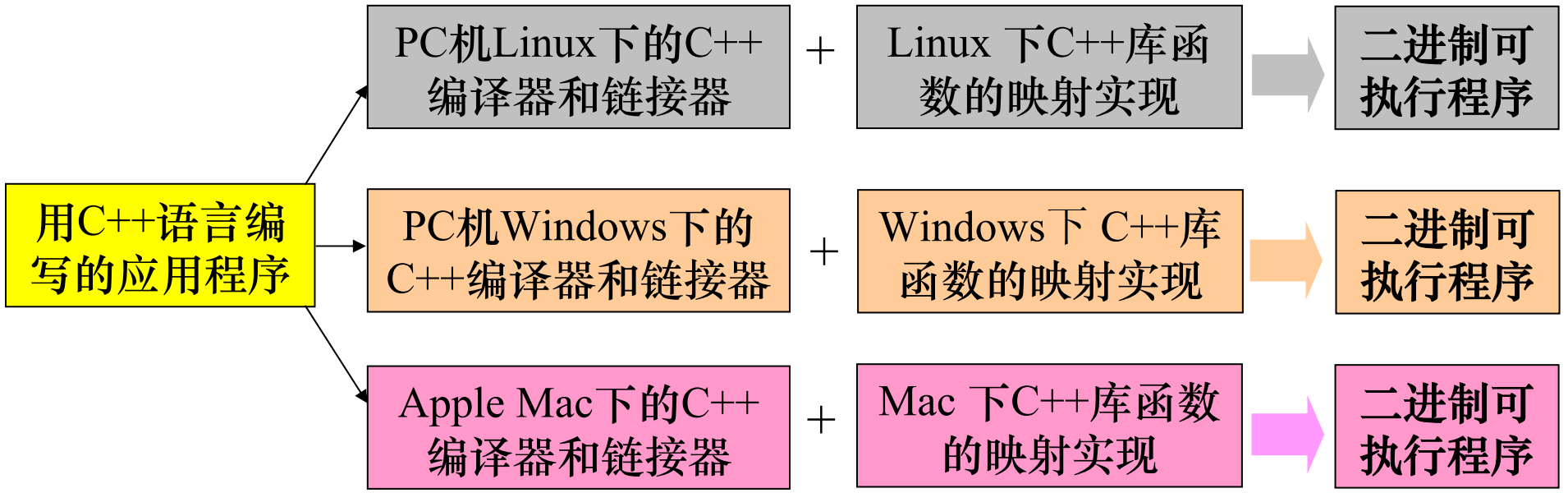
```
malloc(size_t size)  {
    HANDLE h = GetProcessHeap( );
     Return HeapAlloc(h, 0, size);
}
```

- 在Unix/Linux 中的Map实现：

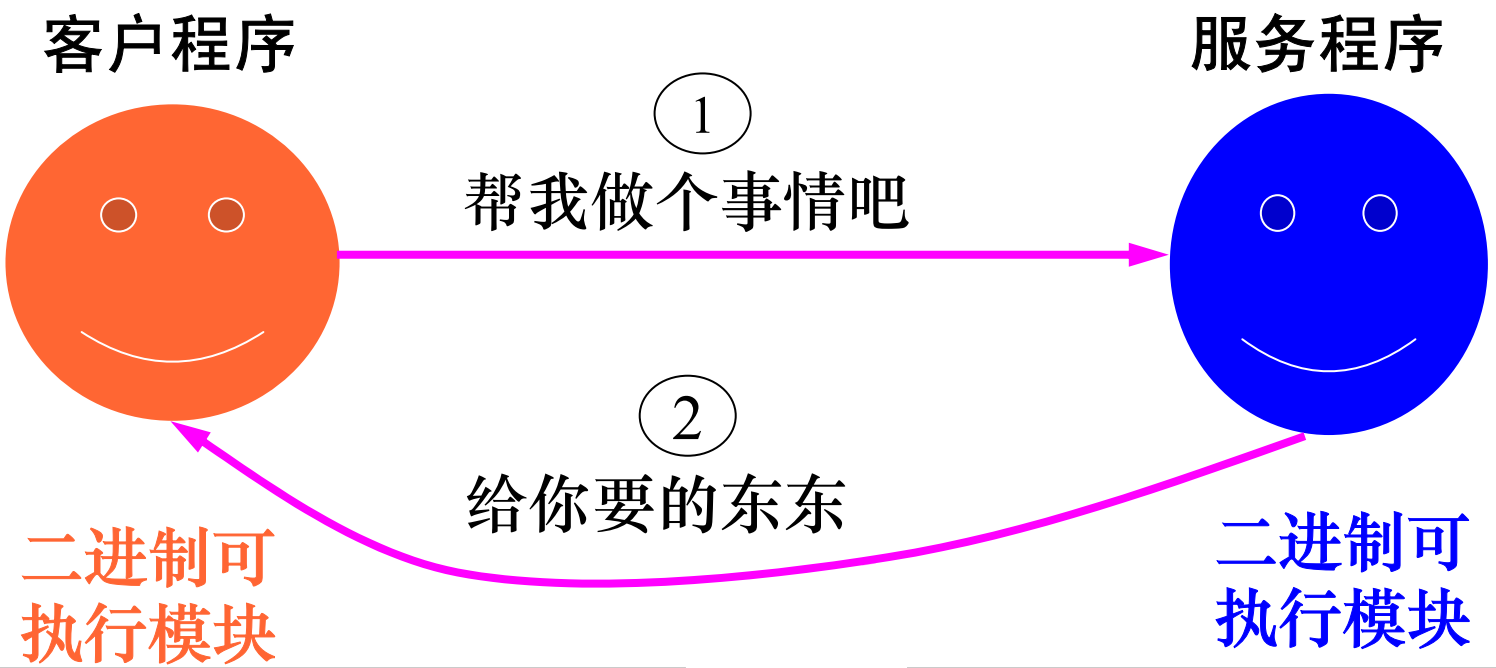
```
malloc(size_t size)  {
    return  sbrk(size);
}
```

高级程序语言实现的通用性特征

注意：这是在**源程序一级实现的通用性**，并没有在**二进制可执行程序一级实现通用性**，因为对于特定的机器和操作系统，都需要使用与其对应的高级程序语言函数的映射实现库，加上与其对应的**编译器**和**连接器**把源程序编译成能在其上运行的**二进制可执行程序**。



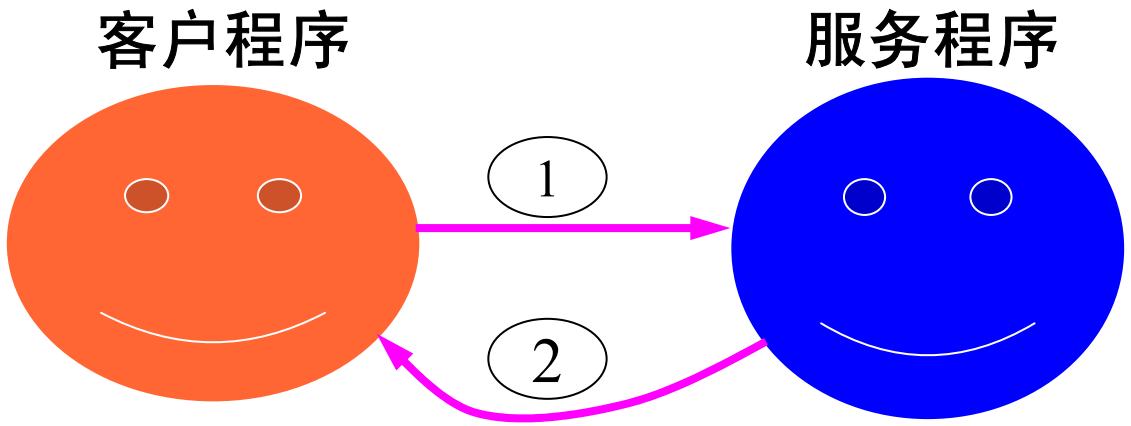
出路3：软件的互操作



这个功能模块我不需要自己开发了，采用拿来主义解决！轻松！快捷！省钱！

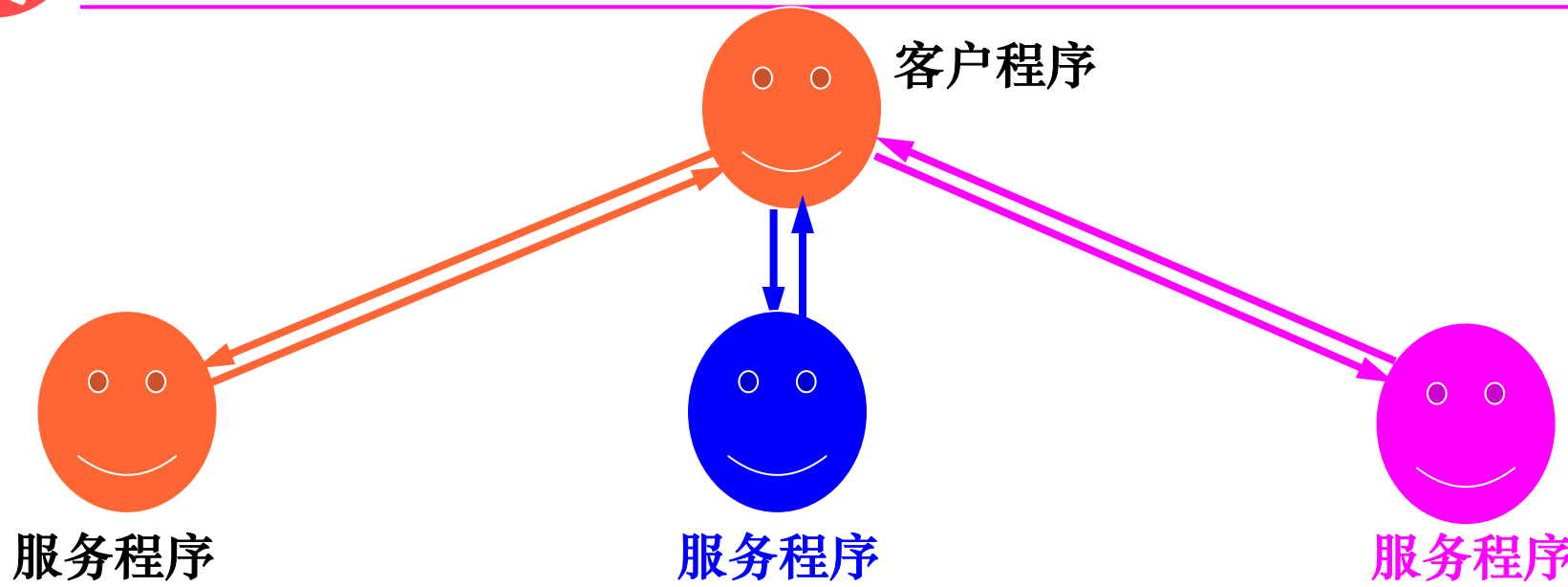
使用我的人越多，我赚的钱就越多，我就更来劲把它做得更好！守住市场不流失！吸引更多的客户！

软件互操作中的问题



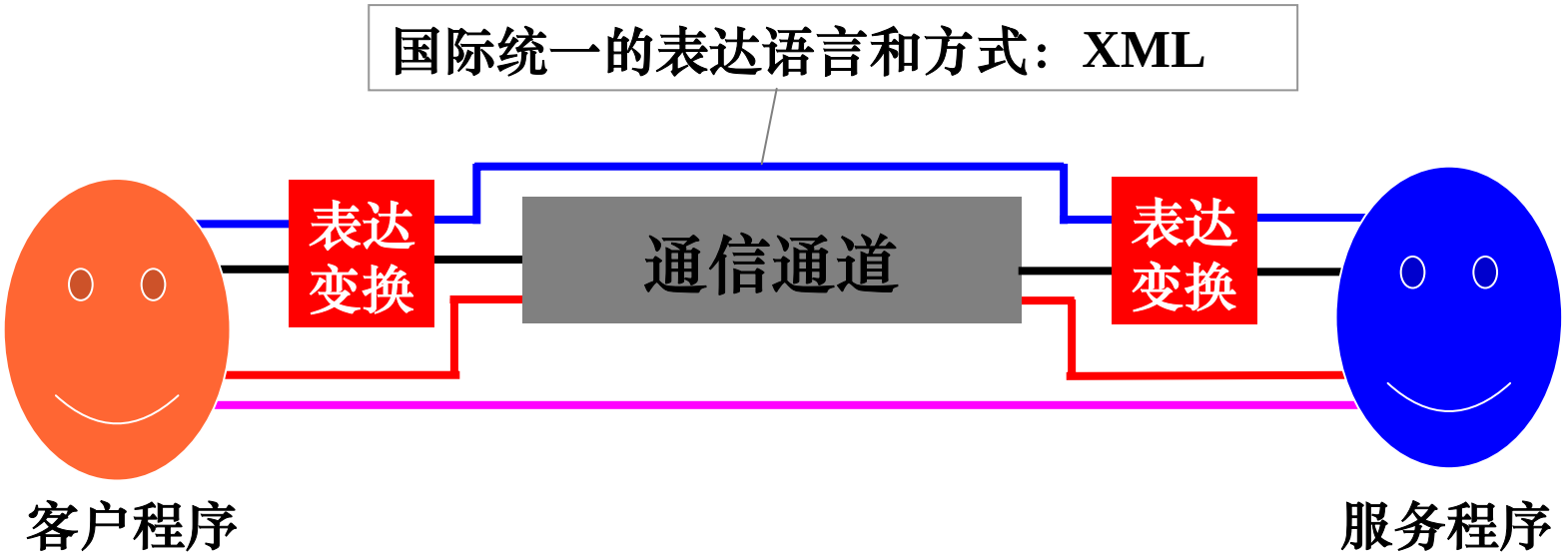
- 我讲汉语 ↔ 我讲英语；
- 我用的是中国式交谈模式 ↔ 我用的是西方式交谈模式；
- 我用的是中国式的邮寄地址表示方式 ↔ 我用的是西方式的邮寄地址表示方式；
- 服务程序，你在哪里？我怎么联系你？
- 我对你发来的东东不认识？不是我们使用的语法规则！

软件互操作中的3种场景



本地人互动	本国人互动	国际互动
当面交谈	电话交谈	电话+翻译交谈
互操作问题容易解决	要有通信基础设施	要有通信基础设施；
简单、便捷、高效！	要能发现对方	要解决语言、方式问题；
	要付费！效率低！	成本更高！效率更低！

软件互操作要求具有灵活应变性



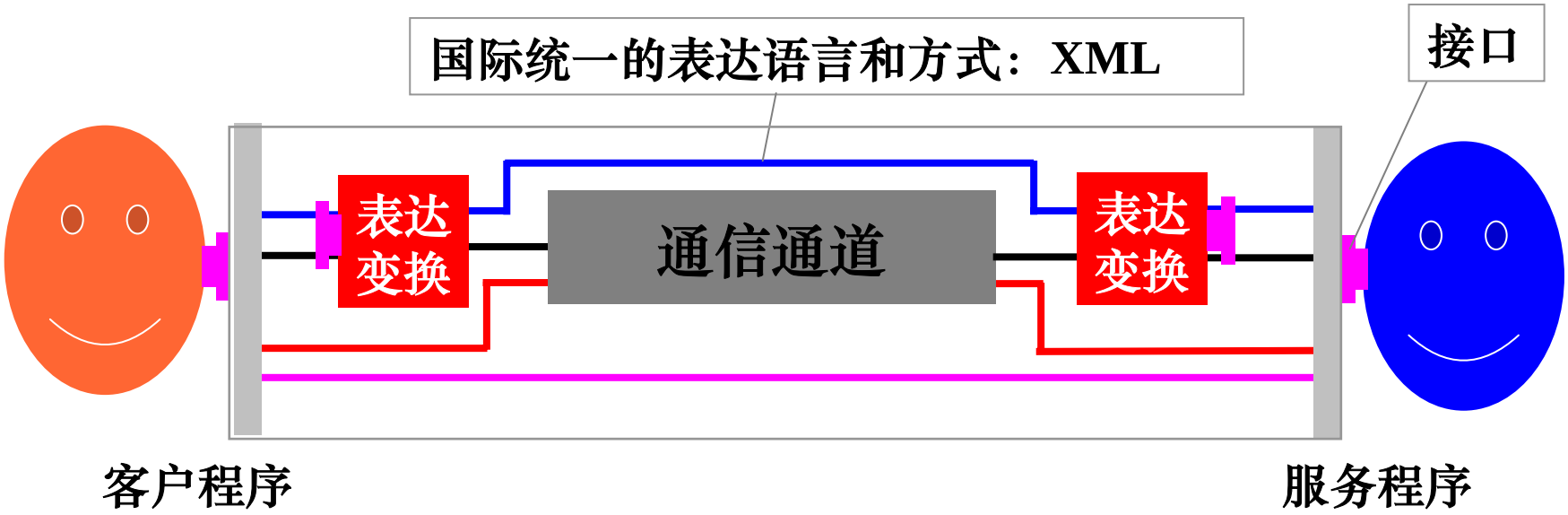
客户程序和服务程序：

- 如果使用不同的数据表达(包括表达方式，排列顺序)，则需要表达变换；

  例如，客户程序是使用C++写的程序，而服务程序是用Java写的程序；

- 如果不在同一机器上运行时，则需要通信通道；

软件互操作中要求具有的不变性

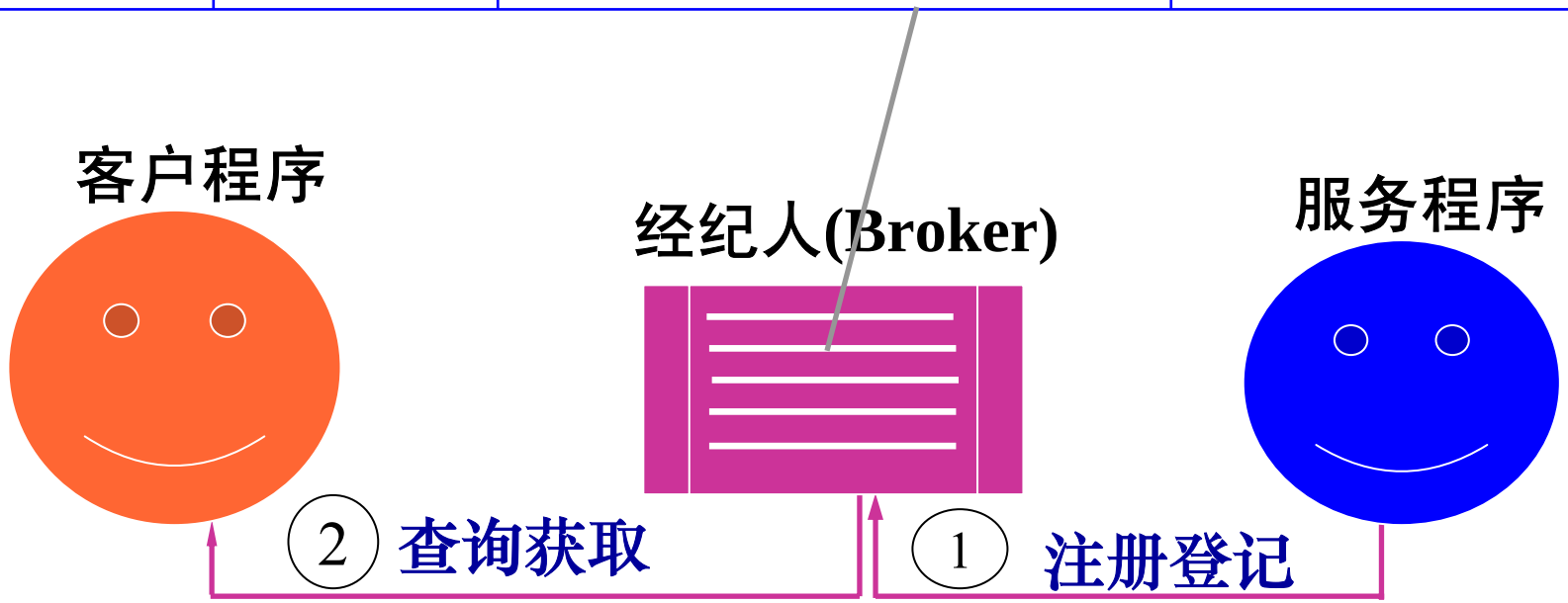


对于客户程序和服务程序开发的设计者和程序员来说：

- 表达变化和通信通道都应该是透明不可见的，即不需考虑的，以便只需关心功能的实现，根本不用考虑各种形式的互操作，保证原有方式的继续有效，放低应用软件的实现门槛，简化设计和编程；

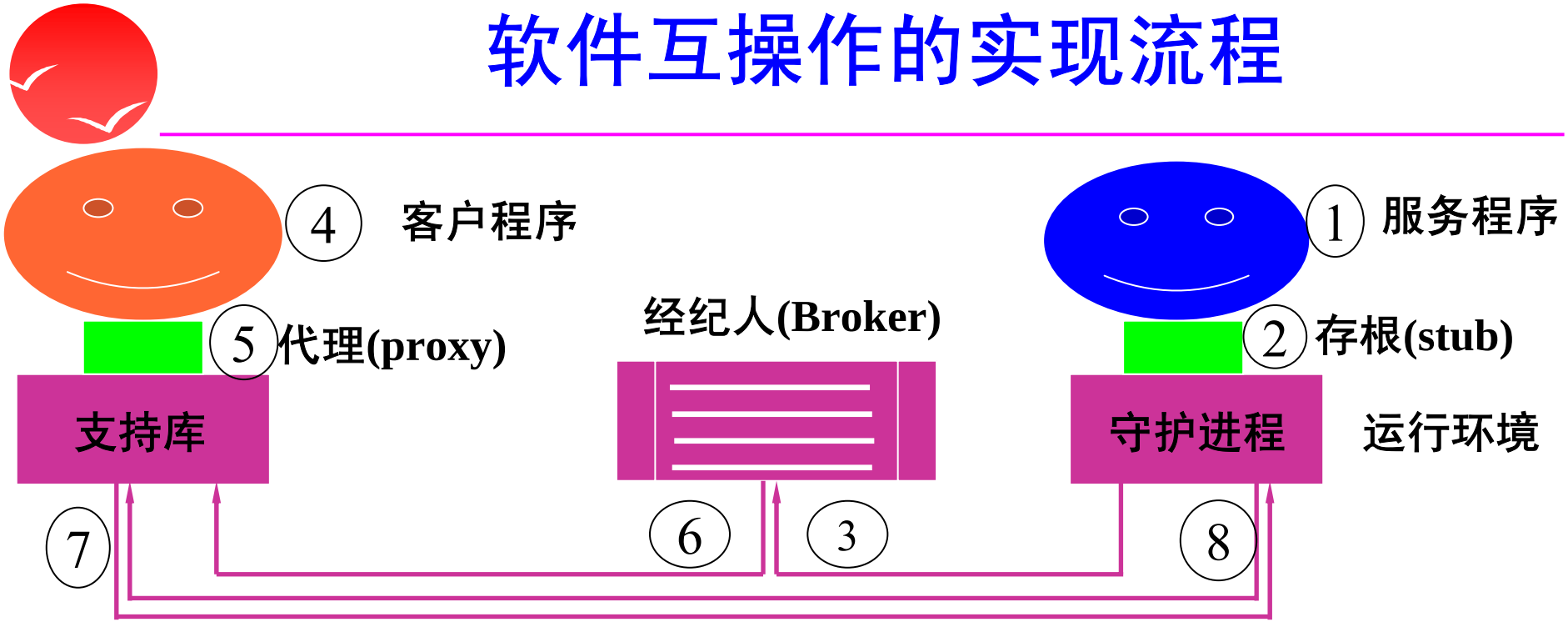
软件互操作中服务程序的发现问题

名称	功能	地址	接口
短信服务	短信发送	152.10.198.25:80/sm.dll	Send(…)
文件服务	文件云存储	152.10.100.111:52/store.dll	Save(…); Read(…);



服务程序的**注册登记**，以及客户程序对服务程序的**查询获取**，**都需要有标准**；这样**注册登记程序**和**服务查询获取程序**就具备**通用性**，对任何经纪人都能**对接，交互**；

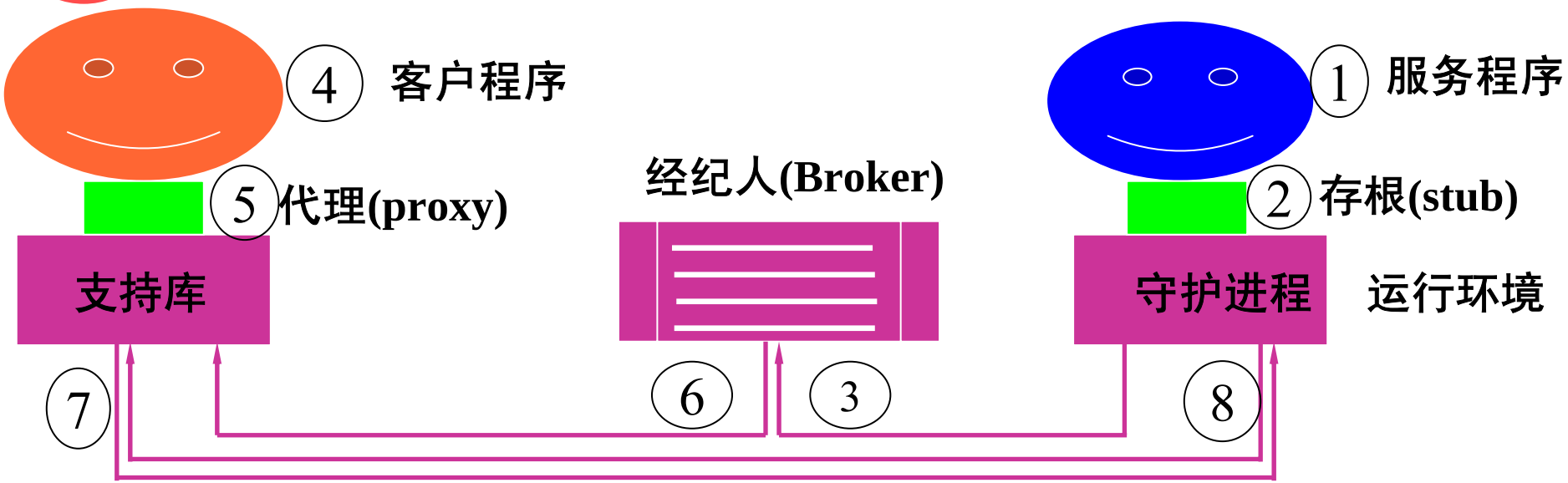
软件互操作的实现流程



先定义互操作接口，例如: int OpenCloudFile(string FileName, int OpenMode);

- **代理**：把客户程序调用的接口名及其参数**打包成请求消息包**，然后**调用支持库的通信接口**将其发送给服务方的守护进程，然后等待对方的结果，再将其返回给客户程序；
- **守护进程**：监听客户的请求，基于**客户请求的服务程序**，**加载运行服务程序**，把客户请求消息包传给存根，把存根返回的结果再转给客户；
- **存根**：**解析客户请求消息包**，得到客户要调用的接口及其调用参数，然后调用服务程序的接口，将调用结果返回给守护进程；

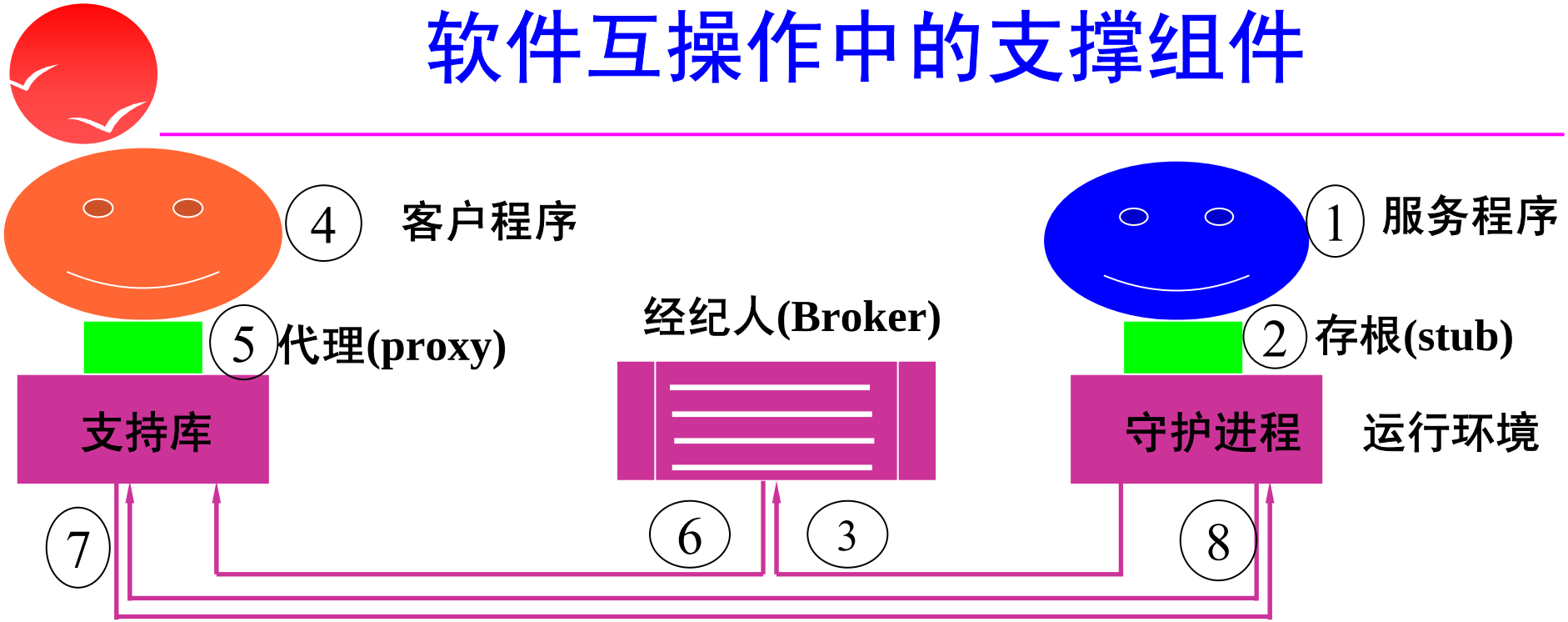
软件互操作中的代理和存根



在给定互操作接口的前提下，例如: int OpenCloudFile(string FileName, int OpenMode)， **代理和存根的功能是固定的**：代理把接口名及其参数打包成请求消息包，发送，等待结果，返回结果给客户程序；存根则解析客户请求消息包，调用服务程序的接口，将调用结果返回给守护进程；

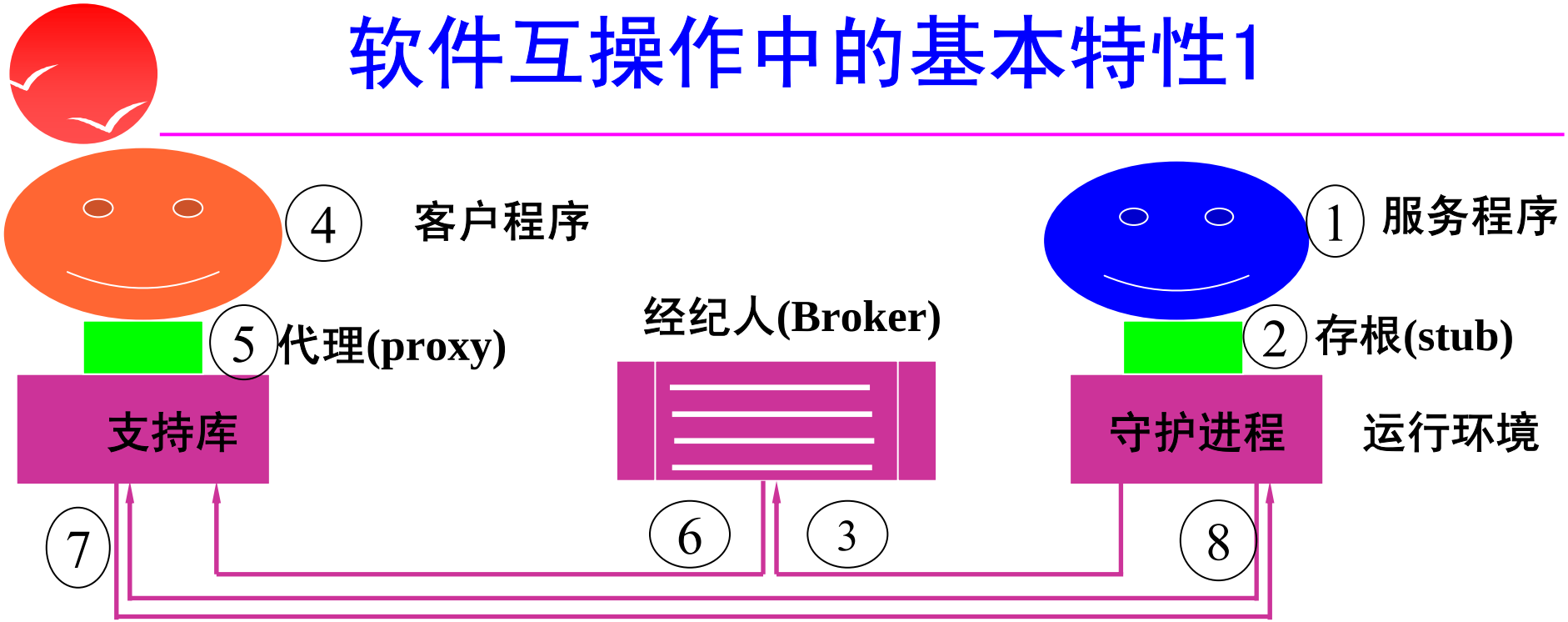
因此，给定互操作接口后，其**代理和存根的代码**完全可以**由开发工具自动生成**，程序员根本就不需要考虑。

软件互操作中的支撑组件



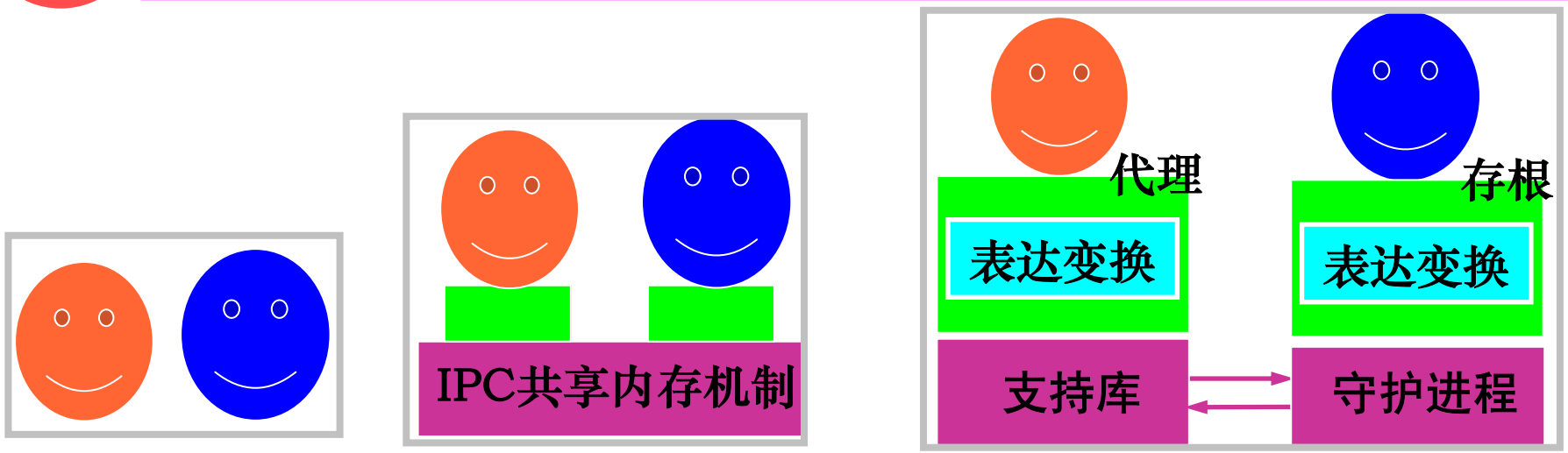
- 运行环境：包括支持库，守护进程，以及经纪人，通常由系统软件提供商提供，例如微软，IBM等；
- 开发工具：作用是基于用户定义的接口自动生成代理代码和存根代码。当客户方编写客户程序时，也将代理代码编译成一个二进制可执行模块；当服务方编写服务程序时，也将存根代码编译成一个二进制可执行模块；
- 客户软件；
- 服务软件；

软件互操作中的基本特性1



- 分离性：客户程序、代理、服务程序、存根、运行环境都是独立的二进制可执行模块；
- 透明性：代理、存根、运行环境对于应用程序员（包括客户方和服务方）是透明的，因此，放低了应用程序编程的知识要求门槛；
- 兼容性：历史遗留服务程序，只要为其配备代理和存根，就具备互操作功能，不需要对其做任何修改；

软件互操作中的基本特性2

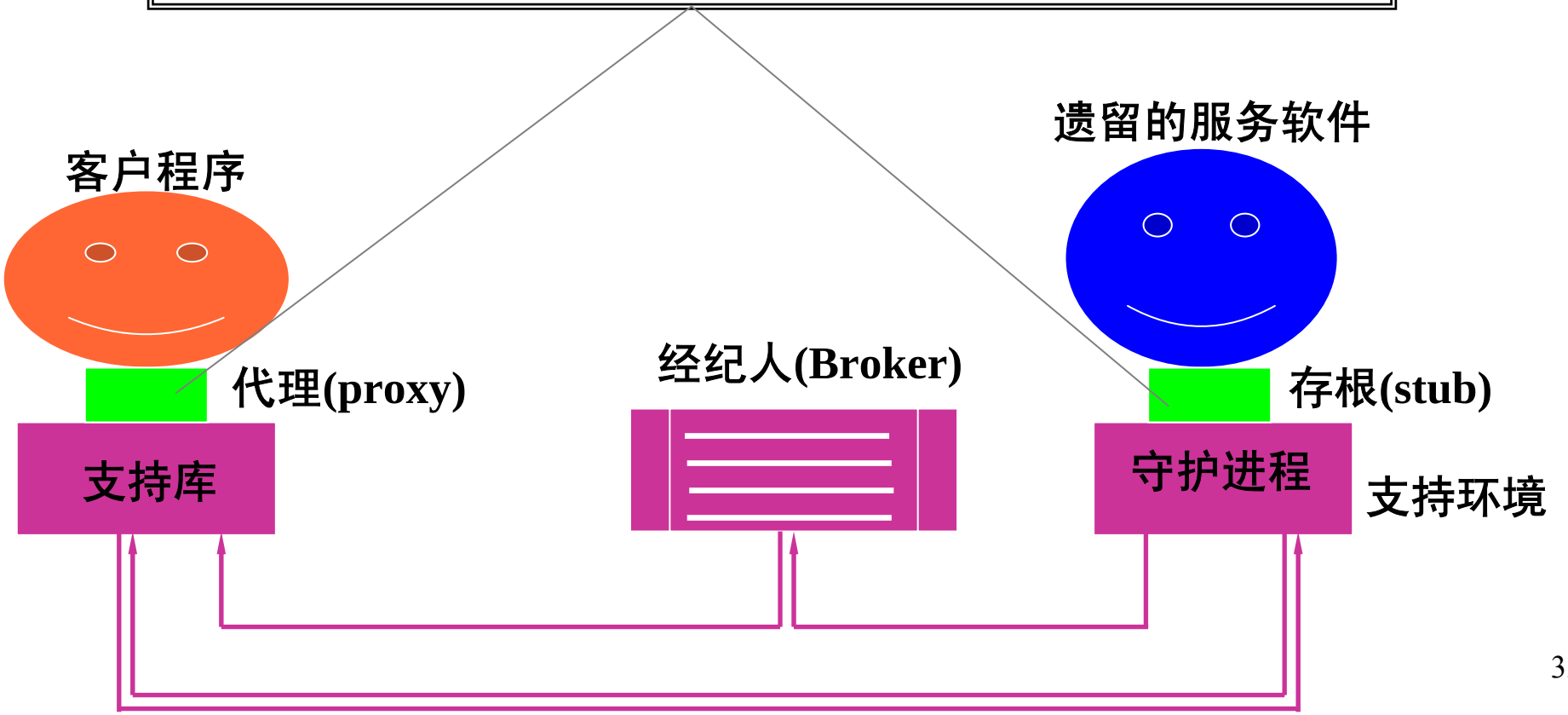


动态性：

- 可以把客户程序和服务程序配置在同一个进程中，直接进行互操作（免去代理、存根、运行环境）；
- 也可以将它们配置在同一个机器的不同进程中，通过代理、存根、IPC共享内存进行互操作；
- 还可将它们配置在不同机器上，通过代理、存根、通信通道，以及表达变换来进行互操作；

软件互操作中的前向兼容性

基于互操作规范，对历史遗留服务软件，只要为其生成代理和存根，就可为其增添互操作功能



中间件技术的含义

运行环境
开发工具

应用软件
的开发

应用软件
的部署

应用软件
的运行

互操作接口定义

客户程序

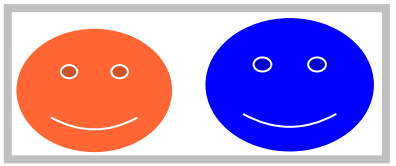
服务程序

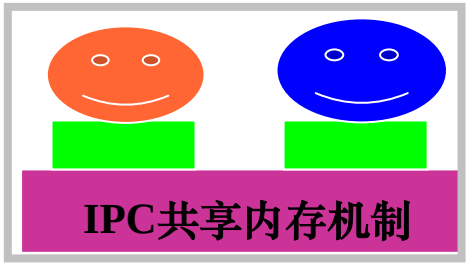
由程序员完成

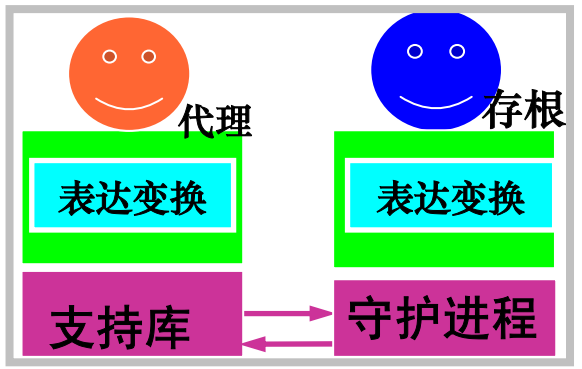
由开发工具
自动生成

代理

存根







运行环境：主要指客户方的支持库，服务方的守护进程；以及经纪人。Windows平台上的注册表就是一个典型的经纪人；

开发工具：编辑器+编译器+连接器，程序员给定接口定义，它具备自动生成其对应的存根和代理源代码的能力；

中间件特性

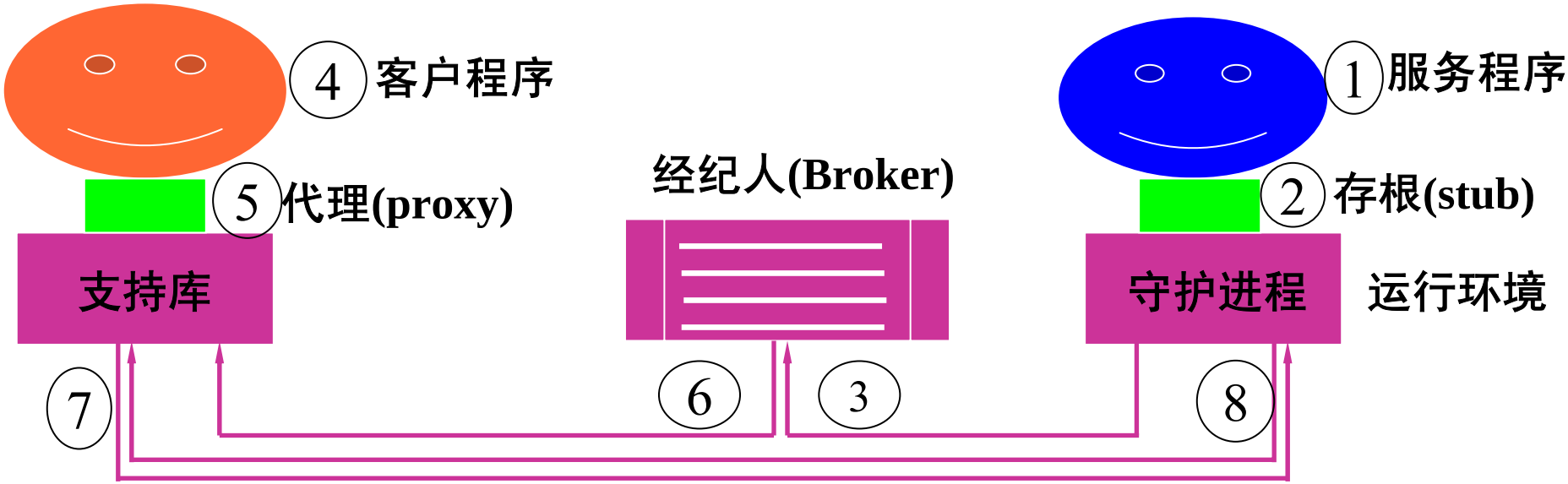
1）二进制可执行程序级的互操作；

2）服务程序可能有多个客户；

3）客户程序和服务程序都会升级进化，无约束条件，可组合性；

4）对应用程序员，原有编程模式可以兼容，依旧只用关心功能的实现；

5）客户和服务之间的契约（即接口的定义）既要具有永恒性，又具有可延伸性；



软件互操作实例

以C语言来展示：

第一步：定义互操作接口：以头文件printf.h 方式提供；

printf.h 文件:

//软件互操作接口：

bool printf (char * p，int i, double j, double k);

服务程序：接口函数的实现

真实的实现函数在printf.c文件中，由服务方程序员开发:

```
printf.c：
bool printf (char * p，  int i, double j, double k)
{
    int factor = atoi(p);
    if (factor *j > I*k)
        return true;
   else
        return false;
}
```

服务方程序员将其编译链接生成：./original/printf.dll

代理程序：由客户方的开发工具根据接口定义自动生成

printf_proxy.c：

```
include <MiddlewareEnvironment.h>
include "printf.h"
bool printf(char * p, int length, double width, double area)   {
    string ServiceId = GetServiceId FromConfigurationFile(  );
    Pipeline = CreatePipeline(ServiceId);
    Pipeline.AddDllName("printf_stub.dll");
    Pipeline.AddFunctionName("printf_stub");
    Pipeline.AddString(p );
    Pipeline.AddIntParameter(length );
    Pipeline.AddDoubleParameter(width);
    Pipeline.AddDoubleParameter(area );
    Pipeline.Call( );
   Pipeline.WaitForResult( );
   bool ret = Pipeline.FetchBoolValue( );
   Pipeline.Close( );
   Return ret;
}
```

定义在

紫红色的这些函数都由运行环境部分的支持库（例如微软windows下的Ole32.dll)提供并实现；

注意：代理的名字和服务程序的名字要求取成一样；

客户方程序员将其编译链接生成：./proxy/prinf.dll

存根：由服务方的开发工具根据接口定义自动生成

```
printf_stub.c
include <stdio.h>
include "printf.h"
bool printf_stub(char * parameterpackage)   {
    char *p;
    int length;
   double width;
   double area;
    p = AbstractString ( parameterpackage);
    length = AbstractInt ( parameterpackage) ;
    width = AbstractDouble ( parameterpackage) ;
    area =   AbstractDouble ( parameterpackage) ;
   bool ret = printf ( p, length,width,area);
   return ret;
}
```

紫红色的这些函数都由运行环境部分的支持库（例如微软windows下的Ole32.dll)提供并实现；

Printf函数由服务程序printf.dll提供并实现；

服务方程序员将其编译链接生成：./stub/prinf_stub.dll

服务方守护进程(Damon)的实现

```
void _ServiceThread(    )   {
   unsigned char szDllName[128], szFunName[128], pBuffer[4096]
   Bool   (*pfn)( unsigned char *) = 0;
   Pipeline = CreatePipeline ("PortId=80");
   while (1)   {
         Pipeline.WaitForPortRequest (  );
         Pipeline.FetchDllName ( szDllName);
         Pipeline. FetchFunctionName ( szFunName);
         Pipeline. FetchParameters ( pBuffer);
           HINSTANCE h = LoadLibrary ( szDllName);
           if (h)  *(FARPROC* )&pfn = GetProcAddress ( h, szFunName);
           If (pfn)  int result = pfn(pBuffer);
         Pipeline.AddIntValue ( result );
          TriggerEvent ( hReturnEvent);
   }   //while
}
```

紫红色的这些函数都由运行环境部分的支持库提供；

操作系统API函数：将一个动态链接库加载到进程中；

操作系统API函数：基于函数名在一个已加载的动态链接库中，找到它的起始地址；

调用此函数；

守护进程由微软、IBM之类的运行环境提供商提供，编译链接生成 ./server.exe

客户程序

Client.c:

```
include <math.h>
include <stdio.h>
include  “printf.h”

main( )   {
  int i=12;
  double j =3.6;
  printf(“Area(%d,%f3.2)=%f3.2”,  i,  j, i*j);
  return;
}
```

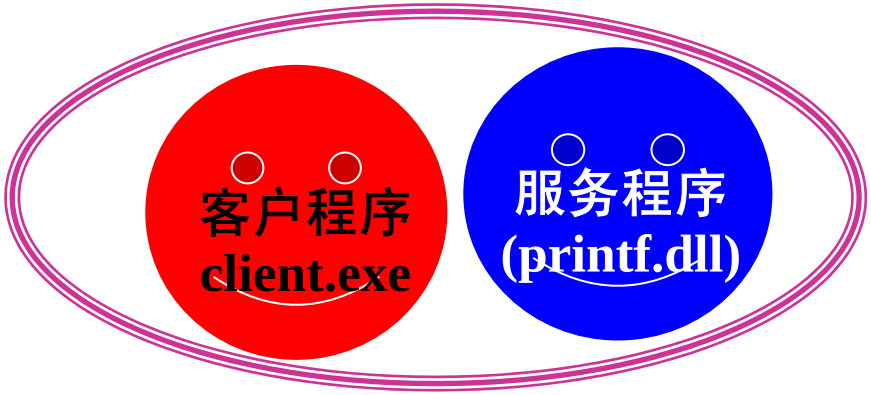
该函数由服务程序提供实现，客户程序去调用它，即使用服务；

客户方程序员将其编译并和printf.dll 链接生成：./client.exe

应用程序部署实例

情形1：　同一进程中同族情形下的互操作（指客户程序和服务程序使用相同的高级程序语言，同一机型和操作系统下的开发工具开发而成）：

系统安装人员将**客户程序员**开发的client.exe文件，以及**服务方程序员**开发的./original/printf.dll文件，拷贝到**用户机器**上的应用程序运行目录C:\app\下，即完成了部署，**用户**即可打开和运行client.exe文件；

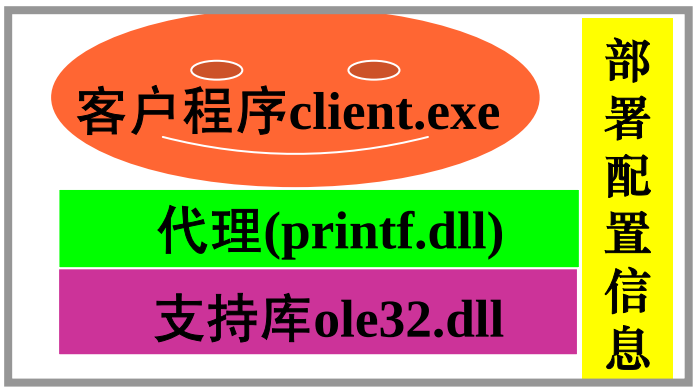


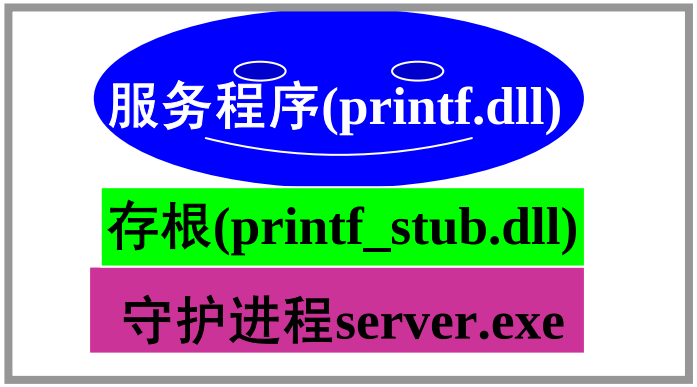
应用程序部署实例

情形2： 不同机器上的软件互操作：

系统部署人员将服务方程序员开发的服务程序./original/printf.dll文件以及存根printf_stub.dll，拷贝到服务器机器上的服务程序目录C:\ServiceApp\下；在服务机器上启动运行环境提供的守护进程server.exe

系统部署人员将客户程序员开发的客户程序client.exe以及代理./proxy/printf.dll文件，拷贝到用户机器上的应用程序运行目录C:\app\下；在用户机器上的注册表（即经纪人）配置服务信息：服务器的IP地址，服务端口号，服务名；





Java对代理和存根的处理

软件互操作中，只要给出了接口的定义，代理和存根的源代码就可由开发工具自动生成。基于这一点，Java更进了一步，**把代理和存根的自动生成这一工作从开发工具移入到Java虚拟机（JVM）中，由JVM实时生成**，于是代理和存根这两个概念对于程序员就完全透明了，既看不到代理和存根的源代码，也看不到代理和存根的二进制可执行模块了，更加放低了程序开发的门槛。

应用程序的部署工作并不透明，客户程序和服务程序的安装，依然需要**系统部署人员**来进行配置，为客户程序配置**服务程序的地址**；

互操作的范围

有三种形式：它们都要考虑**服务程序的标识，动态适配**

- **同一进程内不同模块之间的互操作；**

  可以**直接**完成；

- **同一机器中不同进程之间的互操作；**

  要求进程之间通信，不用考虑数据编码**Data encoding**；

- **不同机器之间模块之间的互操作；**

  要求机器间通信，要考虑不同**Data encoding**之间的转换，因此须要**中间翻译**；还要事先约定**传输协议**；

Data encoding（数据编码）

- **ASCII码**：上世纪60年代，美国标准，字符用1Byte表示，最高bit为0；仅只能表达127个符号，对于英语，完全足够。
- **Unicode编码**：现在的规模可以表达和容纳100多万个符号 ；
- **UTF-8**：互联网上使用最广的一种**Unicode的实现方式**；
  - 对于单字节符号，UTF-8编码和ASCII相同，实现了兼容；
  - 对于n字节的符号(n>1)，1th Byte前n个bit都设为1，第n+1个bit设为0，next Byte的first 2 Bits为10，Rest bits为这个符号的unicode码；

例如：“严”的unicode是4E25（100111000100101），其UTF-8编码需要三个字节：“11100100 10111000 10100101”，转换成十六进制就是E4B8A5。

我们看到每个网页，其开头都指定了编码:encoding=“UTF-8”，浏览器看到这一点，便按UTF-8来解析网页后面的内容

传输协议

- 共享内存IPC；

- TCP/UDP协议；仅仅只是byte序列；

- HTTP协议；包含了结构、编码、语义(keywords)

通用的HTTP解析工具进行内容解析；

- SOAP协议；进一步引入了编程中的对象、接口概念；

完成传输的编程API；

中间件技术的优点

- ✓使用中间件技术之后，软件的开发的**重心**从**功能实现**变成了功能**组件的组合**；解决了**成本问题、开发周期问题、质量问题，提高了软件的利用率**。
- ✓软件的功能的扩充**不再需要重新编译和链接**，可以在**二进制级完成软件功能的扩充**；因此谁都可以对一个软件进行扩充，进行剪裁；
- ✓总之中间件技术使得软件**从封闭变得开放**，变成可为**别人服务**，**自己也可以扩充**，取得了**互利互赢**的功效。

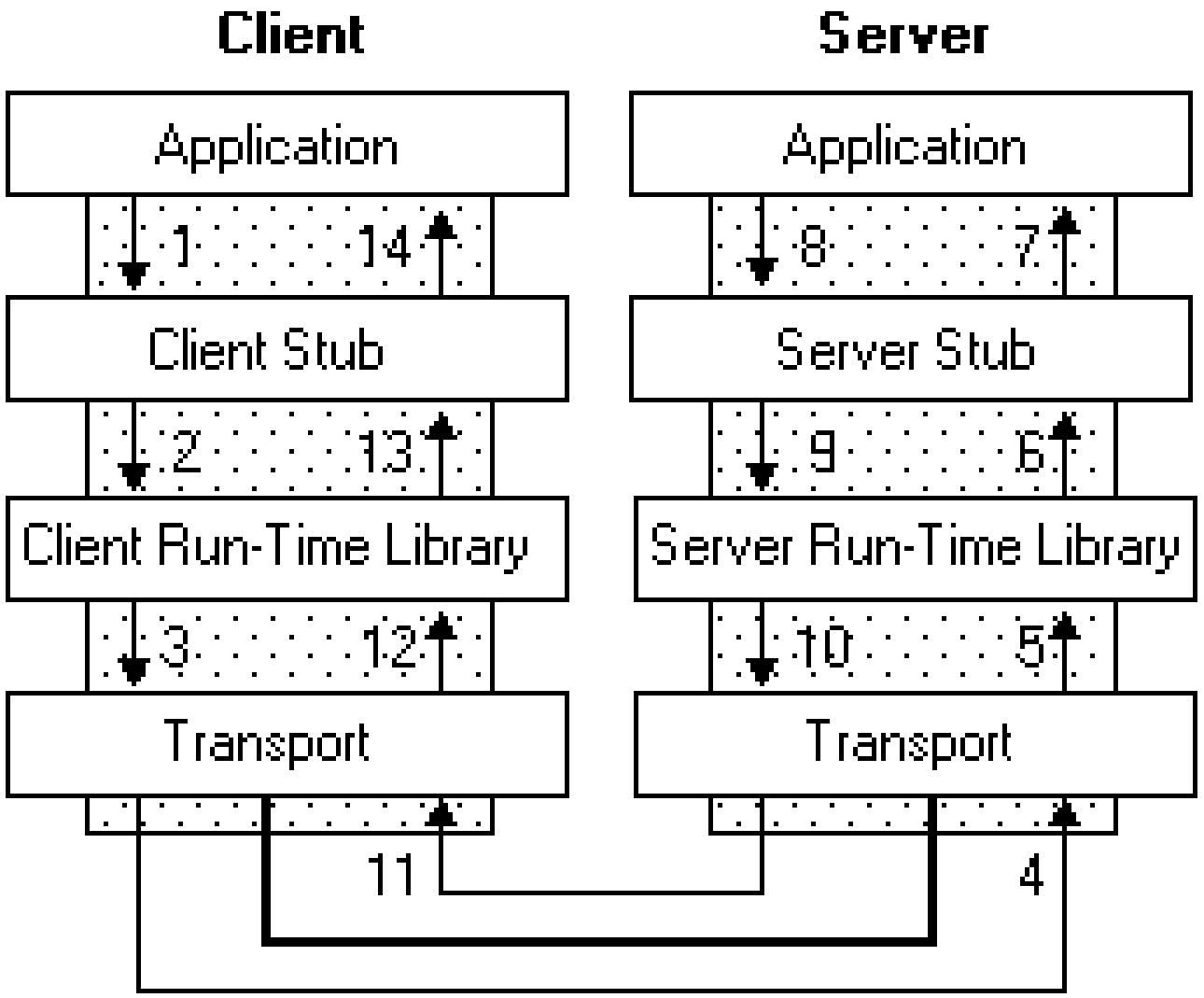
中间件技术的发展历史

- 第一代：DEC RPC;
- 第二代，OLE2, VBX （1993）;
- 第三代：CORBA(1994)，COM(1995)，EJB(1999);
- 第四代， Web Service 和XML(2001);

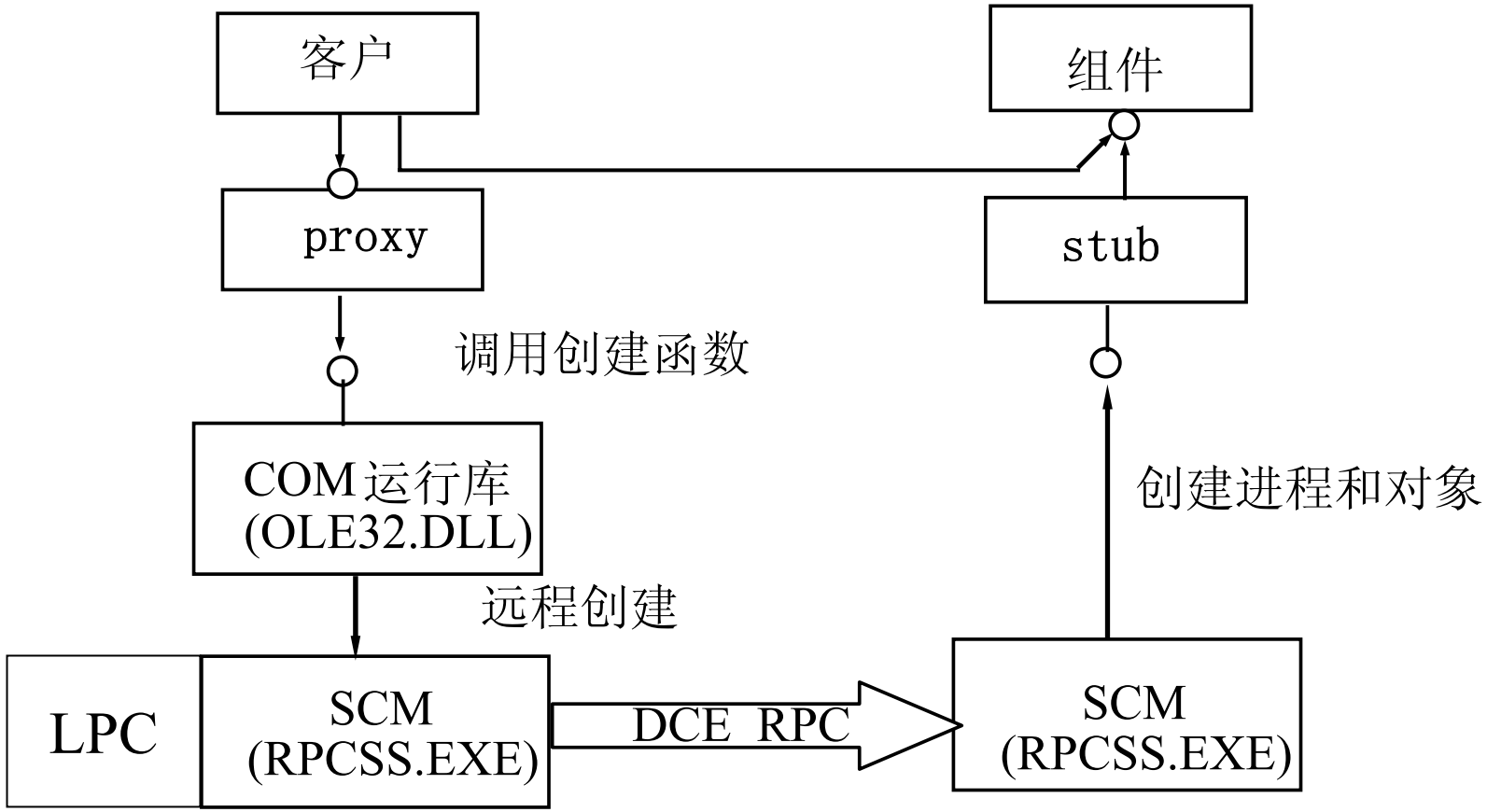
RPC

NDR format

传输协议



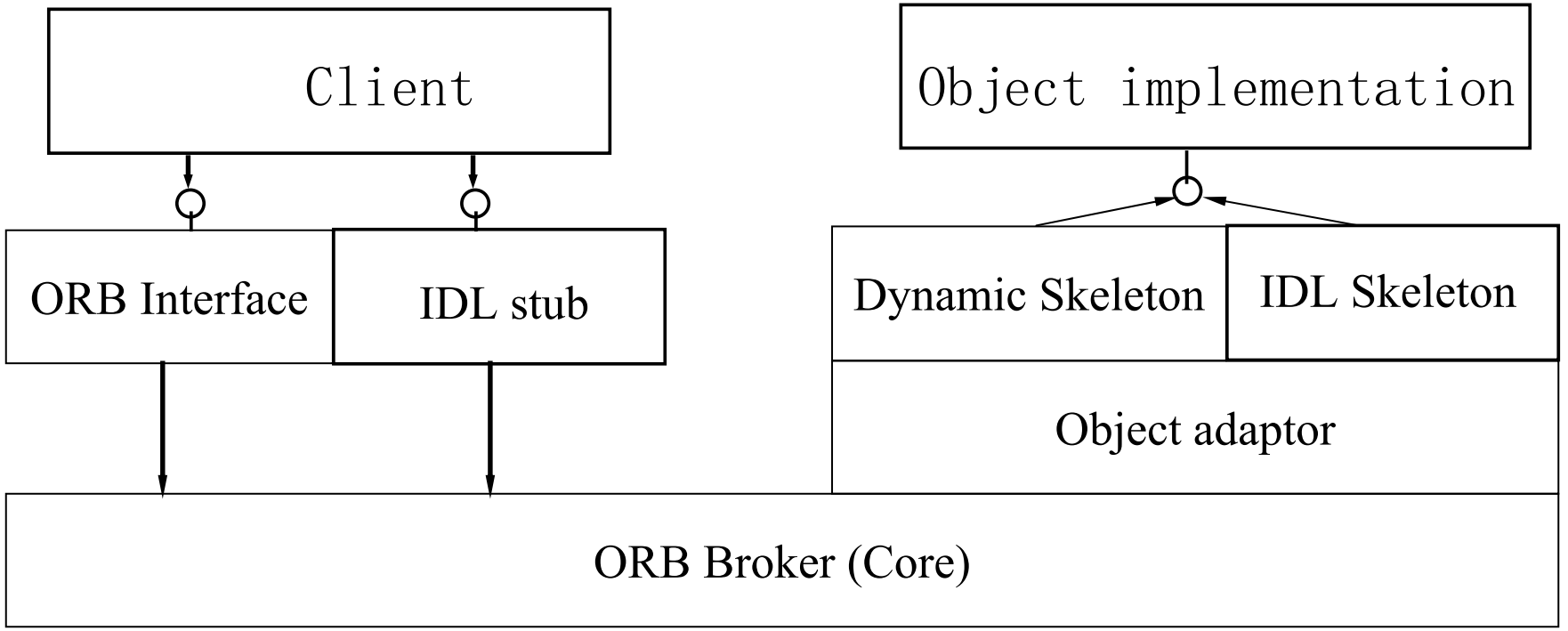
COM



ORPC (TCP + NDR)

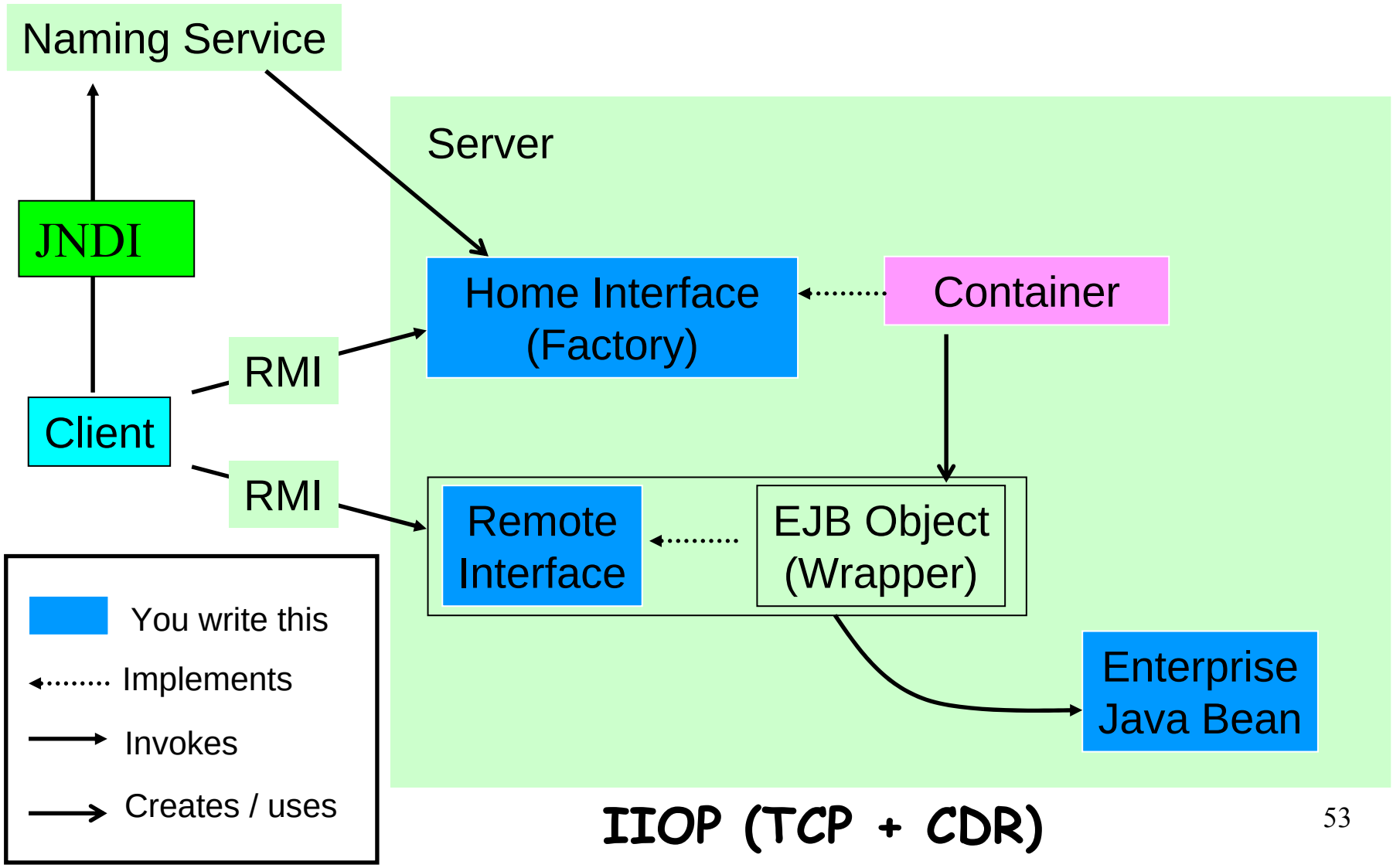
Network, shared memory(IPC)

CORBA



IIOP (TCP + CDR)

EJB



IIOP (TCP + CDR)

透过现象看本质

上述4种中间件技术，从外表来看，它们似乎完全不同，表现在外表不同，用词不同（例如，有的stub和proxy，都叫stub，在CORBA中stub叫skeleton，Broker在EJB中叫Naming Service)。其实，**它们的本质是完全一样的**,都是说互操作问题。造成这种现象的原因是当时没有标准，各自发挥，各自提出见解，自然会外表不同，用词不同。因此**学习一定要抓本质，不要看字面。**

软件人才的基本专业素质

在竞争愈来愈激烈的软件业中，软件人员都必须具备：

- 精通**面向对象技术**(交互接口、事件驱动)和UML建模技术；
- 快速掌握系统架构的能力(**互操作/中间件**)；
- 快速掌握程序语言的能力(操作系统、**数据结构**、**算法**、视窗)；
- 快速**查找资料**的能力、**英语**能力；
- **精致化**的开发能力；
- 也需要了解软件技术发展的**规律**，以便规划自身的发展方向。

软件技术的学习方法

- **软件行业的发展，终会像发展了上百年的汽车工业一样，从百花齐放，到走向成熟**。最后只剩下少数的IT企业（IBM，微软，google，oracle)，从开发工具、服务器、到支撑软件从源头提供成套产品。大部分软件人员将从事装配性、定制性、维护性工作，或者信息服务工作。
- 面对**程序语言**、**数据库**、**组件模型**、**平台**，以及**Framework**等不断冒出的新技术(新名词)，很多人似乎陷入了**永远学不完新东西的梦魇**。
- 不过，如果你仔细思考这些技术的本质，就会感悟它们都不过是对**代码和数据**的精致整理而已。抓住了本质，就不会陷入永无止境的学习梦魇，就会在学习中**游刃有余，如鱼得水**。

章节小结1

- 软件包括两部分：代码和数据；
- 软件技术发展从百花齐放，逐步过渡到邦联式的共识和统一；
- 发展的源动力：最小化成本，最大化收益、最强的竞争实力；
- Internet的迅猛发展，使得系统集成/整合成了必经之路；
- 解决软件危机问题的途径：编程的自动化，源代码的通用，软件互操作；
- 中间件技术就是二进制可执行程序级的软件互操作/系统集成/系统整合/软件组装技术；解决了软件改版需要重新编译和链接的问题；

章节小结2

- 中间件技术使得软件开发的重心从功能实现变成了功能组件的组合；解决了成本问题、开发周期问题、质量问题，提高了软件的利用率。
- 中间件技术就是软件互操作技术；
- 中间件技术包含运行环境、开发工具、应用程序三方面的内容；
- 中间件技术涉及到三方：客户，服务，经纪人；
- Web Services 和XML是最终中间件技术；
- 中间件技术使得系统部署成为一项重要工作，自然会形成一支很大的从业队伍；

章节小结3

- 应用程序之间的整合变成了首要问题。
  - Internet/Intranet和Web的影响非常深远；
  - 原有的百花齐放式的软件技术和程序语言，没有想到互操作问题；
  - 使用一种松耦合(loose coupled)构架，即邦联模式（既有独立性，又有可装配性），并借助标准的沟通方式和数据表达来整合/集成已有系统是必经之路。
- Web Service 是跨平台的框架标准，XML则是跨平台的数据表达标准；
- 整合应用系统须要自动化工具以及一个整体构架（SOA: Service-oriented Architecture）——（开发工具，和系统支持环境）。

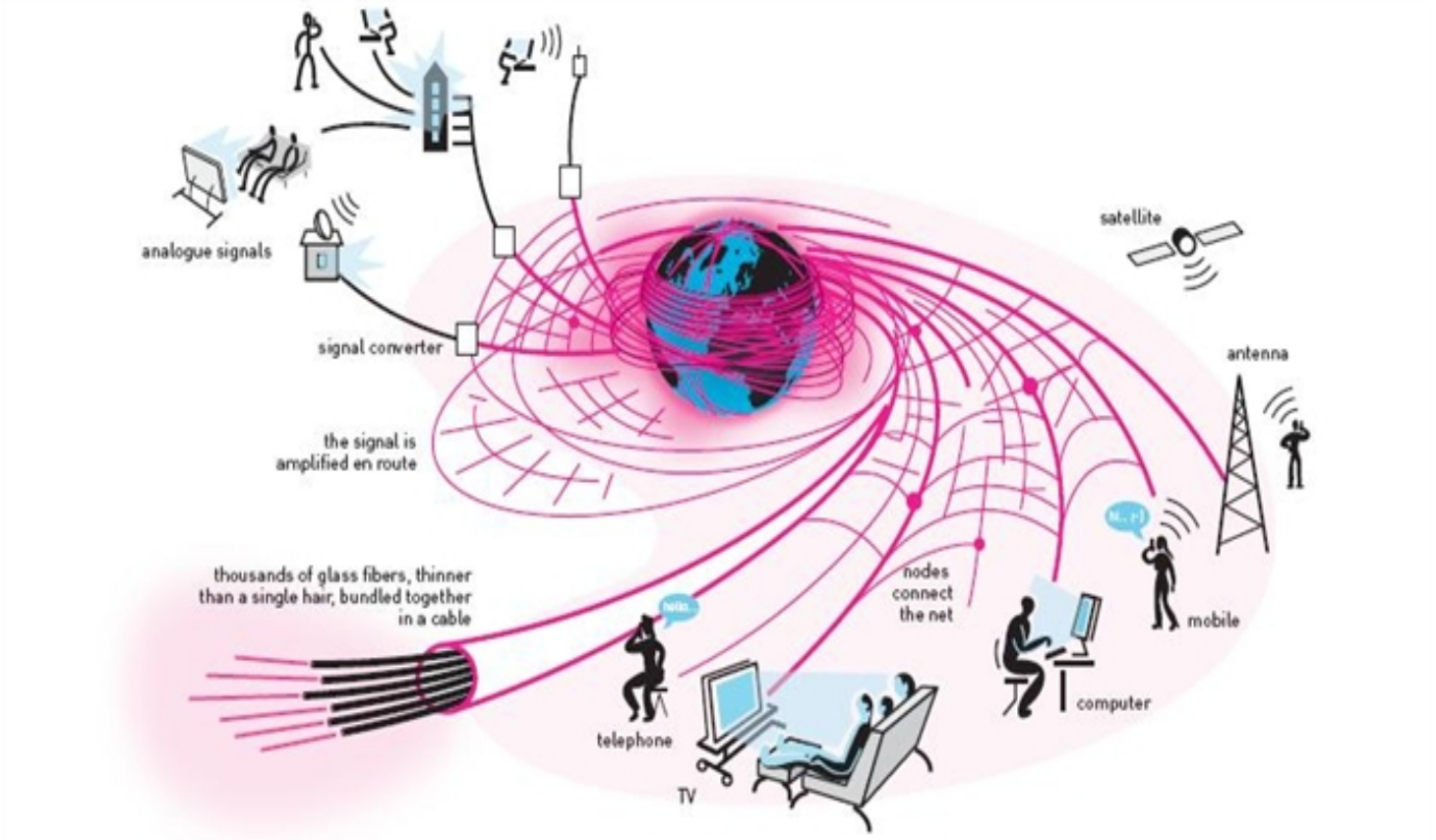
课后思考的问题

- 请查阅资料，了解JAVA中的软件互操作的具体实现方法，重点关注**接口定义**，**代理**，**存根**，以及**数据表达**，以及**通信**的实现框架。
- 请查阅资料， 思考**C++**, **Java**， **C #** 有什么相同和差异？差异背后所持的观点是什么？

课程后续内容

- 同一进程中软件互操作的实现技术；
- 面向过程编程的本质及其在互操作中面临的问题；
- 面向对象编程技术的本质及其在互操作中面临的问题；
- 面向组件编程技术的本质及其在互操作中面临的问题；
- 面向服务编程（Web Services和XML）的本质；

谢 谢！


analogue signals
signal converter
the signal is
amplified en route
thousands of glass fibers, thinner
than a single hair, bundled together
in a cable
telephone
TV
nodes
connect
the net
computer
mobile
satellite
antenna