维普资讯 http://www.cqvip.com

计算机科学 1998Vol. 25 №. 5

可互操作对象

Interoperable Objects

蔡希尧　刘西洋

（西安电子科技大学软件工程研究所　西安 710071）

摘　要　Object-oriented technique is now recognized as the main stream of software development, what is the next step? We are convincing that the interoperability of object-systems may be one of the most important problems. In this paper, the four main aspects of interoperable objects are discussed, including the standards, the supporting of operating systems, integrating compound documents, and the integration of distributed objects with WWW.

关键词　Interoperable objects, Object-orientation, Interoperability, Distributed object computing, CORBA, COM, SOM, OLE, Open Doc, WWW, JAVA

一、引言

基于网络的软件密集型系统正在成为当前信息系统的主流。对象技术现在已经被认为是软件开发的主流技术，在异构的网络环境下，如何有效地实现对象计算，要解决的一个关键问题就是对象的互操作。

在异构环境下对象的可互操作性是指对象之间可以在满足安全约束的条件下以统一的方式自由地交互作用，实现协同计算，而不会受到诸如体系结构、操作系统、程序设计语言、地址空间、网络协议以及软件开发环境等因素的限制，这是分布式处理和网络计算的基本要求。按照互操作的层次可以将对象互操作划分为 OOA&D 层次的语义互操作和传统 OOP 层次的计算互操作，前者的一个典型实例为下文即将论述的 CORBA 元对象设施 MOF 和 OMG 一致建模语言 UML。按照互操作的实现方式可以将其划分为基于对象传递的互操作（不仅传递数据而且传递代码）和传统基于数据传递的互操作（仅仅传递数据而不传递代码），前者集中体现在 Java Applet 以及因 Java 语言的提出而被广泛关注的流动代理（Mobile Agent）[1]，流动代理的典型实例为 IBM 东京试验室正在研究的 Aglets WorkBench（URL：http：// www. ibm. co. jp/trl/aglets/white-paper. htm）。

本文将就对象互操作有关的主要方面，包括对象互操作标准，操作系统的支持，复合文档的集成，以及分布对象和 WWW 的集成等展开讨论。

二、可互操作对象的发展概况

可互操作性是开放系统的基本特征之一。可互操作对象是指那些超越常规边界、需要和其他对象交互作用的对象。工业界需要一种可分布的、可互操作的面向对象机制，使得不同的面向对象和非面向对象的应用可以联结到一起，所需要的应当是基于主流的面向对象环境，并且是二进制的（语言中立的）标准，本质上能支持分布处理。

80 年代，分布式对象计算逐渐引起人们的关注，特别是客户/服务器计算模式的出现，它已成为网络服务和分布式计算的范型。现在，客户/服务器模式已经普及，于是，异构网络环境下分布式对象计算中的对象互操作问题便被提到日程上来，成为主要的研究课题。到目前为止，总的来说，可互操作对象是一个正在努力达到的目标，技术还不成熟。

要达到互操作，首先需要两个方面的支持，这就是标准和操作系统；此外，需要有相应的集成技术。所以，从八十年代后期开始了标准的制定，接着，Microsoft 和 IBM 先后在各自的操作系统中加入互操作的支持技术，相继取得了成果。在标准方面，主要有 OSF（Open Software Foundation）的 DCE（Distributed Computing Environment），ISO 的“开放分布处理参考模型”（RM-ODP）[2]，和 OMG（Object

维普资讯 http://www.cqvip.com

Management Group)所开发的 CORBA(Common Object Request Broker Architecture)等[3];操作系统中的支持性技术有 Microsoft 的"部件对象模型"(COM,Component Object Model)[4],IBM 的"系统对象模型"(SOM,System Object Model)[5]等;集成技术范例有 Microsoft 的 OLE 和 CL Lab 的 Open Doc 等[6-7],这些标准和支持技术,为对象互操作开辟了道路,提供了有力的支持。它们的共同特点是:

(1)支持客户/服务器工作模式;

(2)以对象部件为提供服务的基础,这里,部件是指抽象的具有特性和功能的单元,用面向对象或者结构化程序设计语言写成;

(3)以中间件作为达到互操作的基本工具;

(4)以接口来定义中间件,客户通过接口向对象提出服务请求,采用专门的"接口定义语言";

(5)提供公用设施以支持服务;

(6)语言中立(二进制兼容)。

九十年代中期以来,以 Java 语言的推出为标志,可互操作对象的概念内涵被进一步深化。由 Java 引入的以同构虚拟机网络统一异构网络的概念方法,必将成为以网络为中心的对象计算的基本理念。在同构的 Java 虚拟机网中流动 Java applet 正在引入与传统对象消息传递机制(基本上局限于传递数据)互补的对象行为传递机制,对象互操作机制的内涵由此而深化。同时,由 Java 虚拟机、CORBA 对象请求中介器(ORB)、DCOM 中继服务器、分布事务处理/消息队列中间件、Web 浏览器、Web 服务器、数据库服务器和应用服务器,以及 HTTP(Hyper Text Transfer Protocol)、IIOP(Internet Inter-ORB Protocol)、RPC(Remote Procedure Call)等协议构成的软件网络,正在成为可互操作对象的基本运行环境。

三、标准

在前面所列举的标准中,目前在工业界广泛支持的是 CORBA。CORBA 是 OMG 开发的。OMG 的中心任务是建立基于对象技术并且是商业上可行的一个体系结构和一组规范,用于集成的分布式应用。其首要的目标是要达到:在分布异构环境下基于对象部件的重用、可移植性和可互操作性。为达到这一目的,OMG 通过接口和协议规范来定义一个对象管理体系结构(OMA),以支持基于分布互操作对象的应用。OMA 如图 1 所示。

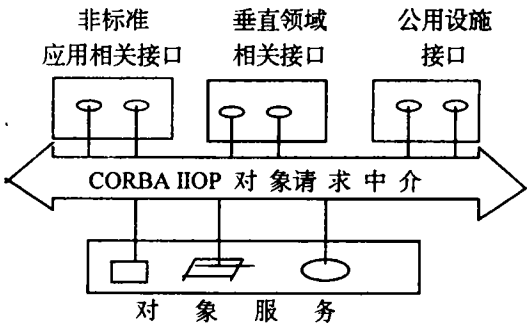

图 1 OMG 的对象管理结构 OMA

CORBA 建立在 OMA 之上,于 1991 年 10 月开始发行,以下几个方面是构成 CORBA 的主要成分:

(1)对象模型:处于 OMA 的底层,描述对客户有意义的概念,如对象的生成和标识、请求与操作、类型与特征,这些描述带有专用性;此外还描述与对象实现有关的概念,如方法、执行机构、激活,这些描述是建议性的,以允许不同的对象技术在实现上有最大的自由。在模型中,对象是按请求动态地生成并消除。对象可以具有任何正常的关系类型,如子类型或父类型关系、继承关系。对象模型是强类型化的,不提供多态机制,但服从 CORBA 的产品允许自由地使用多态。

对象模型中定义两种执行语义:at-most-once:如果所请求的操作成功,恰好执行一次;如果出现异常,最多执行一次。best-effort:只有请求的操作,不返回结果。

(2)对象请求中介 ORB:其作用是透明地为对象请求找出路径,准备对象的实现以接受请求,与提出请求所需的数据进行通信。对位于本地或远方的请求,ORB 能透明地予以响应,不需要客户关心服务对象在什么地方,以及有关对象的通信、激活和存储的机制。这为异构环境下分布对象的互操作奠定了基础。ORB 由接口给予定义,可以有多种实现。

(3)接口和接口定义语言 IDL:接口是一个客户可以向一个对象提出请求操作的集合和这些操作的参数。接口由 IDL 说明。IDL 提供一种与程序设计语言无关的说明对象属性和操作的方法。IDL 服从 C++的词法规则,并对标准 C++的预处理以完全的支持。IDL 的文法基本上是 ANSI C++的子集。

接口与实现严格加以分离。接口库(Interface Repository)则提供在运行时有效的 IDL 信息形式的持久对象的服务。

ORB 的接口和通信协议用于在接口之间传送

维普资讯 http://www.cqvip.com

请求，对所有的 ORB 都是相同的，与对象接口或对象适配器无关。ORB 接口分成三类：①对于所有 ORB 都需要的操作；②对于某种特殊类型的对象需要的操作；③对于特定风格的对象实现所需的操作。ORB 加上接口定义语言 IDL 的编译、接口库和多种对象适配器，构成一组客户的服务。

(4)对象适配器：是 ORB 所提供的一个对象实现访问服务的主要方法。ORB 通过适配器提供的服务包括：生成并解释对象访问，方法调用，交互的安全性，对象和实现的激活或闲置，对象访问到实现的映射，实现的登记。适配器的地位如图 2 所示。

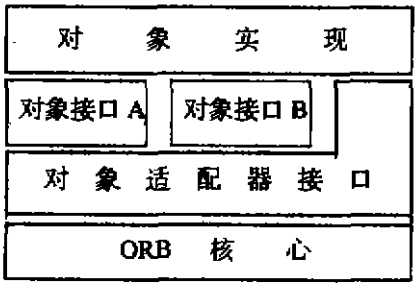

图 2 对象适配器的地位

(5)从 IDL 到程序设计语言的映射及其逆向映射：不同的面向对象或非面向对象语言用不同的方法访问 CORBA。语言映射包括语言特定的数据类型和过程接口的定义。还定义对象调用和对象内的控制线程之间的交互作用。大多数通用的映射提供同步调用。

除了从 IDL 到程序设计语言(如 C/C++，Java)的正向映射之外，还会存在与之相反的逆向映射。需要这种逆向映射的典型场合是：开发者直接用某种程序设计语言而不是 IDL 来定义 CORBA 对象接口，并且所定义的接口必须是可互操作的，典型实例为 SOM 2.1 以上版本引入的 DTS(Direct-to-SOM)[5]以及 Java-to-IDL。

(6)对象服务和公用设施：对象服务包括一组执行基本操作的对象，以支持使用和实现对象的基本功能。公用设施提供多数应用所需要的服务，着重于应用层的功能，如打印、邮件、数据库查询、公告牌以及复合文档。CORBA 定义了"公用对象服务规范"(COSS，Common Object Services Specification)[6]，于 1993 年 11 月开始逐步公布，迄今已经被采纳为标准的对象服务包括：命名(Naming)、事件(Event)、持久对象(Persistent Object)、生命周期(Life Cycle)、并发控制(Concurrency Control)、外部化(Externalization)、关系、事务(Transaction)、查询(Query)、许可程序(Licensing)、属性(Property)、时间(Time)和安全(Security)服务。

进一步应当看到，OMG 已于 97 年 11 月在其 OMA 下的公用设施中引入元对象设施 MOF(Meta-Object Facility)[9]和一致建模语言 UML[10]，以实现不同面向对象方法和工具之间的语义互操作，从而将 CORBA 中的对象互操作层次由 OOP 上升到 OOA&D。同时明确地将以元数据、元对象和元对象协议为代表的经典动态对象技术引入到可互操作对象的工业标准之中。这使得 OMA 更趋近于一个完备的对象体系，而不仅仅是中介现有对象系统和非对象系统的接口抽象。

OMA 本身也存在一些不足，主要表现在：

(1)缺乏严格一致的 OO 概念模型及其形式语义，对象服务之间以及对象通用设施之间相互关联不强，相互难以重用[11]。随着新的对象服务和对象通用设施的不断加入，整个 OMA 体系日趋臃肿和庞大，这与 ODP 形成了鲜明的对比；

(2)对于 CORBA 实现过程中所涉及的关键技术缺乏明确严格的定义，造成实现过程中的二义性，最终影响不同 CORBA 实现之间的互操作能力，一个突出的例子是 CORBA 持久对象服务规范[12]，这与 Microsoft DCOM 形成了鲜明的对比。

四、操作系统中的支持技术[4-7]

操作系统的支持是达到互操作的另一关键。为了使用来自不厂家的、在不同的地方和不同的时间写成的二进制部件，保证可以互操作，这必须解决以下几方面的问题：

(1)所开发的部件和别人所开发的部件可以互操作；

(2)尽可能地减少部件之间的依赖关系；

(3)用不同语言写的部件能够互操作；

(4)用单一的模型生成的部件，能在进程内和跨进程(有时跨越网络)运行。

解决这些问题不能以影响软件的性能为代价。Microsoft 和 IBM 在这方面最早提出解决方案，并在各自的操作系统中予以实现。

1. **Microsoft 的 COM**

COM 是 Windows 系列中的一个组成部分，所以使用广泛。它是为部件的互操作而定义的二进制部件模型，是一种底层的部件软件结构，做为高级软件服务的基础，允许各种应用利用不同软件厂家提

维普资讯 http://www.cqvip.com

供的部件来建成。其特点是:(1)与编程语言无关;(2)适用于多种平台,但目前仍主要基于 Microsoft Windows 平台;(3)适用于基于部件的应用及系统的开发;(4)可扩充。

COM 所提供的机制有:(1)部件之间的通信,可以跨越网络边界;(2)差错和状态报告;(3)部件的动态加载。

为达到以上的要求,COM 在它所支持的环境中,在内存设置一个虚函数表,称为"vtable",同时定义了在 vtable 中调用函数的标准方法。任何语言,只要可以通过双指针调用函数的,都可用来写部件,可以和服从 COM 二进制标准的其他语言所写的部件互操作。COM 中的对象称为"部件对象"。图 3 给出部件对象和 vtable 的联结关系。

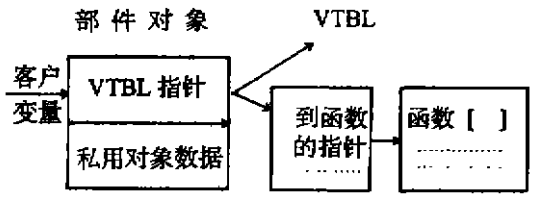

图 3 部件对象和 vtable 的联结

在 COM 中,接口是重要的设施,它是软件部件间的强类型化的契约,有自己的标识,提供一组语义上关联的操作。一个部件对象不允许访问其他部件对象的数据,只有对象的接口对其他对象是公开的。COM 中的接口是一种类型(而不是类或者部件对象),也是一种机制,通过它,客户与部件对象可以通信。客户只能通过指针使用接口,但不能使用指针于对象。部件对象的类能提供多种服务,所以部件对象可以有多个接口。接口是固定不变的。有了这样的接口,部件容易做到与语言无关,而一个客户调用本地服务和远方服务是没有区别的。所有的部件对象的所有接口都必须继承名为"IUnknown"的基本接口,它含有三个方法:

(1)QueryInterface:客户可用来动态地发现一个接口是否为一个部件对象所支持,同时,客户也可以用它从支持接口的部件对象获取一个接口指针。这实质上是一种对象动态认知机制的雏形;

(2)AddRef:当另外一个部件对象复制一个指针用于接口时被调用;

(3)Release:当其他部件不需要使用接口时被调用。

COM 的分布式扩充是 DCOM,它可以让部件对象分布于网络而不是局限于单机。现在,ActiveX 已集成了包括 JAVA 在内的 Internet 技术,使浏览器具有 DCOM 部件对象之间通信的能力,ActiveX/DCOM 以在线方式传送资料,可以适应更为复杂的应用环境。

中继服务器是 DCOM 部件的协调机制,Microsoft 针对这一需要,开发了 MS Transaction Server——MSTS,把 HTTP,DCOM 以及支持 X/Open 的 XA 事务协议的数据库存取界面结合起来,提供完善的分布事务支持。

2. IBM 的 SOM

SOM 是 IBM 所建立的模型,它也是语言中立的对象模型,首先被装在 OS/2 2.0 操作系统的 Workplace Shell 上。SOM 的目的是允许二进制类库对部件软件开发提供与程序设计语言及编译无关的系统级的支持。为达到语言中立,SOM 定义了基于外部过程和简单数据结构的运行时的应用程序接口(API),程序员可以利用这些 API,以通常的面向对象计算模型去生成并使用 SOM 对象,SOM 具有以下的特点:

(1)显式地将元类和反射机制引入到二进制部件模型,并借助 SOM 的分布式扩展 DSOM 进一步将其引入到分布对象计算,从而增强了部件适应变化的能力,同时从根本上扩展了经典动态对象技术的应用范围。SOM API 被表达成以 SOM 对象组成的一个面向对象系统。这样,类也被实现为对象,从而在 SOM 中引入基于元类的程序设计和结构反射机制[5,13-14]。与 ObjVlisp 和 CLOS 类似,SOM 的元体系结构如图 4 所示。

(2)基于微对象核(Mircro Object Kernel)的可扩充体系结构。反射机制和微核结构是建造自适应软件系统的基本方案[15],SOM 将二者有机地结合在一起,其微对象核即为图 4 所示的元体系结构。

(3)对象以二进制方式加以分发和子类化。类库的开发者不需要提供源码而允许用户实现对象的子类化。

(4)对象可跨语言并完全子类化地使用。可用一种语言实现一个对象,用另一种语言子类化,而用第三种语言建立一种应用。开发者可以用他所喜欢的语言对类库中的类加以修改并建立应用,这一语言不一定是原来写这个类时所用的语言。

(5)如果一个类的实现有所变动,不要求变动客户的源码(当码是类的实例的子类或实例的使用),这就可以更换二进制类库中类的实现而不要求客户

维普资讯 http://www.cqvip.com

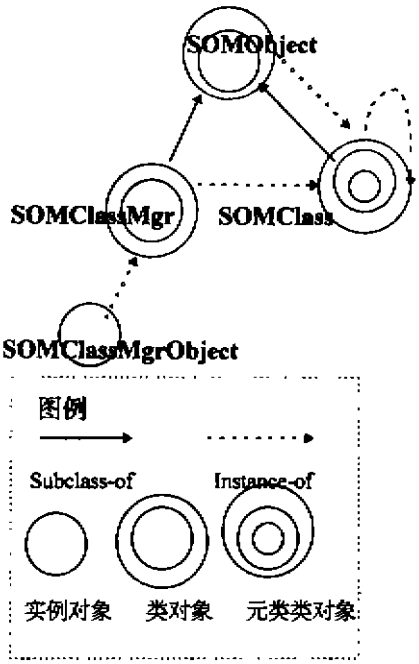

图 4 SOM 的元体系结构

码的重新编译，从而最大限度地支持了部件系统演化的灵活性。

SOM 支持所有与面向对象系统相关的概念与机制，包括继承和多态，支持元类，不同类型的方法的派遣(静态的和动态的方法解析)，动态的类生成，以及用户对方法派遣的中断。

SOM 是通过基于系统定义的过程链接机制来达到跨语言的互操作，也就是说，SOM 所遵循的是由操作系统为所有程序定义的寄存器及堆栈使用约定，不管程序的实现语言是什么。系统链接约定能控制从被调用者给调用者传送返回值的途径。使用系统链接协议，SOM 可以不管执行码所用的语言来派遣方法。所以，任何支持系统的过程调用链接约定的语言都能使用 SOM。

SOM 的类是用 CORBA 的 IDL 来定义的，并为支持所有的 CORBA 数据类型提供一个接口库以支持 CORBA 的功能和程序设计接口。

跨越进程和机器的对象间通信使用 SOM 的分布式扩展(DSOM)。DSOM 是 SOM 的分布式框架，其实现的核心在于借助 SOM 中的反射机制扩充 SOM 定义的方法派遣机制，这样，程序设计者可以在不同的地址空间或不同的机器上透明地把调用方法作用于对象。DSOM 库和 CORBA 规范完全一致，支持所有的 CORBA 数据类型、功能和程序设计接口。

五、部件集成和复合文档

在应用中，可互操作的部件需要集成、构成复合文档。部件集成可以使用一个处于上层的壳或容器作为基本工具，它们可以放置由任意数量的第三方生成的不同数据类型以及相关的功能，并可以方便地加以使用。下面介绍两种典型的集成技术。

1. OLE 2 的集成技术

OLE 是 Microsoft 推出的部件集成和复合文档技术，它把各种类型的功能部件集成为所需的文档，不管这些部件放在什么地方、也不管它们的形状和大小。OLE 所用的集成技术包括以下的主要方面：

(1)结构化的存储：部件集成需要有可用于多个部件共享的二进制的介质，每个存储的部件可以保持其持久的状态。在结构化的存储中，每个二进制阵列分为两类元素：存储和流。存储带有接口 IStorage，流带有接口 IStream。每个元素有一个名。

(2)对象持久性：一个部件在存储和流中所具有的持久性能力是分别通过两个接口 IPersistStorage 和 IPersistStream 的实现来表示的，管理持久性对象的容器程序生成 IStorage 和 IStream 的实例、赋予具有 IPersistStorage 或 IPersistStream 的部件。

(3)持久和智能的名(Monikers)：一个 moniker 是一个封装了名的类型和以这个名进行工作的智能部件，名处在接口 IMoniker 的后面，moniker 的用户在需要使用这个名的时候，向它传送控制。IMoniker最基本的操作是联结对象。OLE 定义并实现了五种 monikers 的类型：file，item，generic composite，anti 和 pointer。

(4)一致的数据传输和 Drag-and-Drop：这是在数据源(叫做"数据对象")和数据使用者(叫做"消费者")之间传输数据和通知数据变化的技术。

(5)通知：使用外部数据的消费者希望知道在什么时候数据源中的数据变了，OLE 通过处于接口 IAdviseSink 中叫做"advise sink"的部件给出通知。

(6)OLE 自动化：OLE 的自动化主要通过接口 IDispatch、ITypeLib 和 ITypeInfo。在 IDispatch 中的一个自动化实体叫做"dispinterface"响应一组专用的定制方法。当生成一个自动的对象时，用对象描述语言(OLD)生成一个文件来定义一个 dispinterface，这个文件经过一个专门的编译器，生成包含任意数量的自动对象和 dispinterface 的所有类型信息

维普资讯 http://www.cqvip.com

的类型库。

(7)OLE 文档:建立在结构化存储、一致的数据传输和 monikers 之上的技术叫做"OLE 文档",这一技术支持复合文档的生成和管理。需要两种部件参与工作:①容器:控制文档并管理文档中各个信息段之间的关系;②复合文档对象:是组成文档的信息段,它们是由服务器提供的。OLE 文档因此是通过复合文档对象把容器和服务器集成起来的方法。这些对象可以通过两种途径被共享:

* 嵌入:整个对象被嵌入于容器中;

* 链接:对象的图形连同指示对象数据实际位置的 moniker 被放在容器文档的快速缓存之中。这两种途径,是 OLE 名称的由来。

(8)In-Place 激活:或称"可视化编辑",是一组接口和协商协议,通过它们,容器和对象把用户接口中的元素合并到容器的窗口空间,还允许对象把编辑工具带到容器中去。

(9)OLE 控制:一个 OLE 控制是一个复合文档对象的自动化扩充,通过 IDispatch 以支持特性和方法,它依靠事件机制,这是一种当某些与控制有关的事情发生时出现的通知。一个控制把不同类型的外部事件转变成有意义的可编程的事件。当这些可编程的事件出现时,事件管理程序可以执行。

2. OPENDOC 的集成技术

OpenDoc 是 CI Lab 开发的基于 SOM 的跨平台复合文档模型,尽管其研究开发目前已基本终止,但其核心技术已被大量地转移到 JavaBeans 之中,因而仍不失为一种极具代表性的复合文档技术。在 OpenDoc 中,文档的内容通过对远方的数据库的查询动态地自动生成。在同一文档内,可以有多种表示,并且还支持文档的压缩和加密。

OpenDoc 用小的、可重用的、可互操作的应用部件按照需要来开发应用,它是生成复合文档的开放式结构,文档中可以包含不同的数据类型。文档可以被编辑、打印、以只读的方式循环。对用户来说,OpenDoc 具有以下的优点:

* 为生成具有一致接口的复杂文档提供统一的、可定制的系统;

* 给出单一的工作区,用户可以在其中剪贴多种类型的内容;

* 如同一个高级的工具箱,可以按照用户的特殊需要剪裁其环境;

* 依靠"Bento"多媒体存储格式来存放文档;

* 备有书写脚本的机制,允许用户显式地书写脚本或者记录他们的动作并转换成脚本,跨越空间和时间进行合作;

* 用拖动第三方的对象标记,用户可以增加新的能力和内容的类型;

* 能够与其他技术及结构交互作用。

OpenDoc 的对象消息传送使用 SOM 的设施。

对开发者来说,OpenDoc 的优点是:

* 使用面向对象技术,多重继承和多态;

* 部件按通用的程式设计,可以重用;

* 使用 SOM 的包装技术生成二进制可执行部件,可以象文件复制那样被替换或更新。

OpenDoc 的结构:OpenDoc 由以下几个主要的子系统组成。

* 壳文档和部件管理器:壳文档提供地址空间,分配事件,以及基本的一致接口源,如窗口和菜单。部件管理器管理部件的显示、编辑和存储;

* 一个几何协商协议;

* 对象存储机制:依靠 Bento 存储系统来保持文档的内容。Bento 能够划分一个存储容器以保持不同形式的数据,能够在一个文件内协调多个数据流。Bento 把文件当做高度结构化的容器,能够保持多个交织的对象。在 Bento 中,一个值是一个任意长度的字节流,带有字节流如何解释的类型。特性则指示值的作用;

* 暴露的事件流:在 OpenDoc 内的事件流对于文档中的部件是可视的,这样,用户的动作可以被记录并被回放。除了键盘和鼠标的按动外,当部件操作时会发出事件,也能够发送事件到流中以改变其他对象的状态或行为。记录对文档所做的变动序列,OpenDoc 能够保持有意义的修改历史以及修改的负责者,以便以后集成时协调;

* 对象调用约定:用 SOM 构造组成文档的对象,这些对象可以用任何语言来写,并使用任何编译,仍然能够交互;

* 对象包装技术。

六、分布对象与 WWW 的集成

Internet 已经成为应用最广泛的网络系统,因此,把分布对象模型集成到 WWW 中去,成为当前分布式对象计算和对象互操作的重要研究课题,二者的集成可以概括地描述为:

WWW + Java + CORBA + DCOM + 数据库 + … = 以网络为中心的对象技术。

经典 WWW 技术是集信息的超拓扑结构表示、

维普资讯 http://www.cqvip.com

呈现、检索、传输、组织和管理的信息共享技术，其计算能力薄弱。CORBA、DCOM 作为分布对象计算的主流技术，迫切需要从传统 Client/Server 计算环境扩展到以 Internet/Intranet 为代表的网络计算环境；Java 本身就是 Internet 程序设计语言，因而其结合是必然的。对象技术始终是贯穿其中的主线索，由此自然导出以网络为中心的对象技术。当前，以网络为中心的对象技术的研究主要从以下三个方面进行：

(1)((CORBA＋WWW)＋Java)＋数据库＋DCOM＋…

即以 CORBA 为出发点，从 CORBA 的角度集成 WWW、Java、数据库以及 DCOM 等其他分布对象技术。代表性实例为 OMG 实施的 CORBAnet (URL:http://corbanet.dstc.edu.au/)；

(2)((Java＋WWW)＋CORBA)＋数据库＋DCOM＋…

即以 Java 为中心，从 Java 的角度集成 WWW、CORBA、数据库以及 DCOM 等技术，代表性实例为 JavaIDL(URL:http://splash.javasoft.com/pages/intro.html)；

(3)(WWW＋Java＋CORBA＋DCOM)＋数据库＋…

即以 HTTP 和 IIOP 为软件对象网络的骨干协议，有机集成 WWW、Java、CORBA 以及 DCOM，并进一步集成数据库和其它相关技术，如 Oracle 提出的网络计算体系结构 NCA(Network Computing Architecture)(URL: http://www.oracle.com.nca/)。这些研究工作必将使得网络环境下的对象互操作得到进一步的完善。

结束语 本文阐述了解决对象互操作的思路、方法和范例。现在，网络计算正在兴起，对象的应用日益普及，因此，对象互操作问题成为当前重要的研究课题，用户期待有更完善的解决方案。

从文中已经看出，对象重用和对象互操作是两个密切相关的问题，关于前者，我们将在另文中讨论。

参考文献

[1] Daniel T.C. et al.,Mobile Agents:A New Paradigm for Distributed Object Computing on the WWW, OOPSLA'96 Workshop:Toward the Integration of WWW and Distrbuted Object Technology, URL: http://www.ibm.co.jp/trl/aglets/ma.html

[2] ISO/IEC,ITU-T,Reference model of open distributed processing, ISO/IEC JTC1/SC21/WG7, ITU-T X.901～904/ISO/IEC 10746,1995

[3] Object Management Group,The Common Object Request Broker:Architecture and Specification Revision 2.0,OMG Technical Document ptc/96-08-04, July 1996,URL:ftp://ftp.omg.org/pub/docs/ptc/96-08-04.pdf.gz

[4] Brockschmidt K.,Inside OLE 2,Second Ed.,Washington:Microsoft Press,1995

[5] IBM Corp.,SOM Objects Developer Toolkit:Programmer's Guide,Volume 1:SOM and DSOM,SOMObjects Version 3.0(First Ed.), IBM Object Technology Products,March 1996

[6] OOPSLA'94 Panel:Development of Distributed and Client/Server Object-Oriented Applications:Industry Solutions,ACM SIGPLAN NOTICES,29(10),Oct. 1994

[7] Orfali R. et al.,The Essential Distributed Objects Survival Guide,New York:John Wiley&Sons,1996

[8] Object Management Group, X/Open, CORBAServices:Common Object Services Specification,OMG Document os/97-07-04. URL:ftp://ftp.omg.org/pub/docs/os/97-07-04.pdf

[9] Object Management Group, Meta-Object Facility RFP, OMG Document cf/96-05-02. URL: ftp://ftp.omg.org/pub/docs/cf/96-05-02.txt

[10] Rational Software Corp.,et al.,Unified Modelling Language V 1.1,OMG Document ad/97-08-11, URL:ftp://ftp.omg.org/pub/docs/ad/97-08-11.zip

[11] Evan W.,et al.,A Situated Evaluation of the Object Management Group's (OMG) Object Management Architecture (OMA), ACM SIGPLAN Notices, 31(10), Oct.1996.OOPSLA'96

[12] Jan K.,et al.,Lessons Learned from Implementing the CORBA Persistent Object Service,Same to [11]

[13] Danford S. et al.,Reflection on metaclass programming in SOM, Same to [6]

[14] Forman I.R.,et al.,Composition of Before/After metaclasses in SOM,Same to [6]

[15] Buschman F. et al.,Pattern-Oriented Software Architecture:A System of Patterns,John Wiley&Sons, 1996