

# 分布对象技术与软件复用

王希辰 (中国矿业大学北京校区)

**摘要:** 分布对象技术利用分布在网络结点上的对象资源进行数据处理, 随着分布式应用软件应用面的扩大, 其开发过程也变得越来越复杂。软件复用是降低软件开发复杂度的重要途径之一, 因此分布对象技术与复用技术在研究内容上呈现出较大的重叠性, 分布式构件概念的出现即是两种技术互相促进、互相结合的结果。本文结合CORBA标准对这一问题进行讨论。

**关键词:** 分布对象 软件复用 构件



## 1 引言

软件复用是指重复使用“为了复用目的而设计的软件”过程。通过复用, 可以控制软件开发的复杂度, 缩短开发周期, 并提高软件产品的质量。由于软件开发模式多种多样, 因此复用的方式也不尽相同, 其中基于构件(Component)的复用是目前学术界与产业界公认的主流复用技术, 与其他复用方式相比, 基于构件的复用更为可行、实用。

分布对象技术重点解决异构环境下的互操作性, 随着中间件技术的逐步成熟, 分布对象的高层开发显得日趋重要, 因而构件思想的引入是十分自然的, 它突破了只能由高级程序员编写分布应用程序的现状。不仅如此, 由于中间件的引入, 构件之间的独立性更强, 为构件组装, 尤其是运行级的构件组装提供了强有力的技术。目前国际上对该方向的研究十分活跃, 各公司、研究机构皆有较大的投入, 目前已经成立了专门的组织CWC(ComponentWare Consortium), 就共同的问题进行研究。

进入九十年代以来, 网络技术得到了迅猛发展, 面向对象技术语言也日益成熟, 再加上二者内在的互补性、协调性, 它们共同促进了分布对象技术的发展。发布对象技术研究分布于网络不同结点上的对象如何进行协作, 并共同完成特定的任务。分布对象技术的核心内容在于对象之间的互操作, 尤其是异构环境中的互操作, 通过中间件的支持, 使物理上分散的计算资源在逻辑上构成一个整体。解决互操作性的关键问题在于制订一套独立于硬件平台、操作系统以及编程语言的规范, 目前存在两套这样的规

范: OMG组织的CORBA规范与微软公司的DCOM规范。CORBA(Common Object Request Broker Architecture)是OMG在OMA(Object Management Architecture)基础之上定义的对象请求代理(ORB)的公共结构。ORB是一种对象式中间件, 它有效地隔离了互操作的双方: 客户对象与服务器对象只需遵守由IDL(接口对象语言)定义的共同接口, 则客户对象即可以透明地激活一个远地服务器对象, 就象是激活本地的对象一样, 而不必关心对象的位置, 使用的编程语言, 具体的操作系统, 或者其他与对象接口无关的信息。CORBA的细节见图1。

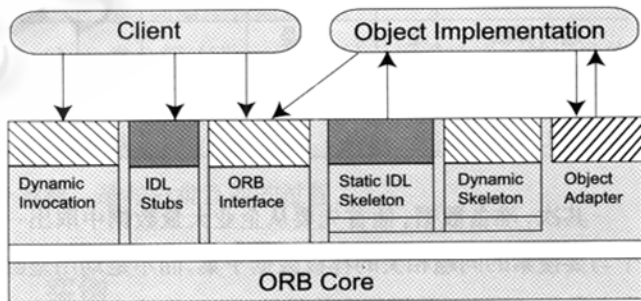


图1 CORBA 结构

COM是微软公司提出的构件标准及实现。它使开发人员可以利用COM的通信机制组装不同开发商提供的构件, COM的核心是一组应用程序调用接口(API), 该接口提供了创建构件、组培训构件的方法。DCOM是微软为支持网络环境而对COM进行扩充的结果, 它的目标与

CORBA类似,都是为了支持不同结点上、不同操作系统、由不同语言实现的对象进行互操作。与ORB不同的是:COM的底层机制仅在客户建立请求关系阶段发挥作用,帮助定位服务器对象的位置,而具体的调用、响应过程则在客户对象与服务器对象之间直接进行。

## 2 基于分布对象的构件技术

应用程序复杂性的提高,要求我们在更高层次上对目标系统进行抽象,并且在更大程度上对以往设计的成果进行复用。构件技术即很好地满足了上述需求。鉴于CORBA更好地体现了分布对象技术,后面的讨论结合CORBA标准展开。

### 2.1 构件特点

构件是用于复用的软件实体。根据复用阶段、复用方式的不同,与之相对应的构件表现形式也不相同,例如分析文档、详细设计、代码实现等。在分布式系统中,构件的最初含义是指分布在各个结点上,相互独立运行,又相互协作共同完成某项任务的程序单元。任意两个构件间的操作是按客户/服务器的方式进行的。发出请求的一方为客户,接受请求并执行相应处理的一方为服务器。客户与服务器互相独立,只有在运行过程中,当客户发出请求时,二者才建立起连接关系。这样,从客户角度看,服务器是早已存在的,之所以要与服务器连接,完全是为了利用服务器所提供的某以特定功能,也正是在这一意义上我们说客户“复用”了服务器。以目前被广泛认可的三层客户/服务器结构为例(见图2),该结构由最终用户程序、应用服务器以及数据服务器组成,它们共同完成用户指定的某项特定任务。

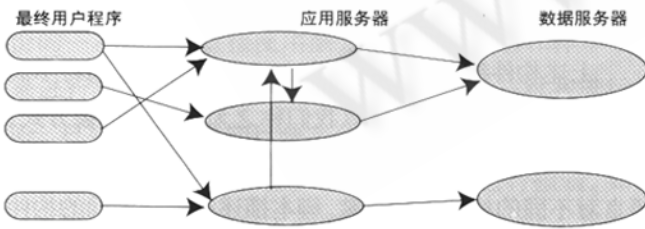


图2 三层客户/服务器结构

从上面的分析不难看出,分布式构件以目标码构件为主,对这种构件的复用不必对代码进行改写、编译等工作,因此复用方式最为直接。实际上,这也正是以往的应用程序对操作系统、编译系统进行复用的方式。所以尽管

这种黑盒式的使用方式使复用者既不能打开它们,又不能根据自己的需要修改它们,但在目前被支持的构件中,它仍然是最重要的构件,COM、JavaBean等支持的也都是该类型的构件。与COM、JavaBean相比,分布式构件更突出的地方在于构件间的连接。COM、JavaBean是在构造阶段与静态的目标代码型构件相连接,这种方式对于本地构件是适宜的,但不适于分布式构件的连接,因为这些构件分布在不同的处理机上。分布式构件之间的连接除了可以采用一般的静态方法建立之外,还可以在运行过程中建立起来,即只有当客户方构件主动向服务器构件发出连接请求时,才建立起连接关系。被连接的构件可以是静态的,在复用者与其连接时由相应的适配器(Adapter)将其启动起来,它们也可以是动态的,即一直处于正在运行的状态,这种形式的构件允许复用者对构件指定共享式或独享式复用。在共享方式下,不同的复用者共享一个构件实体,不同的复用者之间甚至可以通过被复用者建立起联系;在独享方式下,不同的复用者在与构件连接时需要克隆(clone)一个正在运行的构件,其结果是多个复用者分别对应于不同的构件。

### 2.2 构件描述

一个构件必须为复用者提供关于自身的信息。由于CORBA的目标起初定位在解决运行过程中的互操作问题上,因此在对象的IDL中仅提供了关于操作的描述,缺乏对设计过程信息的描述,例如构件的依赖关系、互联关系等,分布式构件不同于其他目标码型构件的关系之处在于构件间灵活的连接方式。为使复用者能够完成对一个分布式构件的使用,如下的几部分内容是必须的。

(1) 属性。这是指构件可以由外部直接操作的特征,属性值可以被读出,也可以被修改。不仅如此,为维护构件内外的一致性,对部分特殊属性的修改还可以作为系统的一种事件向外部通知,触发相关构件中的操作,从而使它们成为一个有机的整体。

(2) 功能接口。这是构件向外输出的可操作方法,也是对构件功能与活动的一种定义。显然,这也是CORBA规范中对象的IDL所定义的内容。

(3) 依赖关系。这个关系指出了当该构件被实例化时它所依赖的构件或接口,这是构件完成其功能所必需的部分,这种关系既可以是其他构件,又可以是其他构件的特定接口。

上述关于构件的描述通常被称为构件的元数据(Metadata),每个构件都拥有各自的元数据。构件必须是可以浏

览的,即可以由工具将各个可复用构件的元数据列出来,使复用者了解到关于构件的必要信息。复用者获取这些数据时,对应的构件不必一定是已经实例化的,因为此时的复用者尚未决定是否利用该构件,完全没有必要在这个时候建立构件的实例。对元数据的存储有两种方式:较直接的方法是将其存储到对象实现描述池中,以作为对象实现描述的补充部分;但更稳妥的方法是将它们放到一个构件描述池(CDR)中(类似于接口池:IR),以增强对元数据的管理。

对元数据的描述语言称为构件描述语言(CDL),它是IDL的超集。

## 2.3 构件组装

### 2.3.1 构件连接

对于源码型构件,相互之间的连接方式相对简单,只需首先对源码进行必要的分析,然后在调用者代码中适当的位置加入对被调用者的调用语句即可。不过在这种方式下,一旦建立起连接关系后就不易进行调整,因为连接关系信息被编译在目标码中,如果欲对其进行调整,必须修改源码,并重新编译。与之相反,分布式构件间的连接必需解决在不改变目标码的前提下加入连接信息的问题,因此其连接机制要复杂得多。一旦实现了连接机制后,构件间的连接方式就变得灵活,尤其是增强了动态配置的功能,使用户可以在系统运行过程中增加新的构件或替换它的构件,并且不影响系统的正常运行。

为复用一个对象的实现,我们经常希望用一个新的实现去替换一个老的实现。在C语言中,函数指针可以作为一个公共变量进行传递,使编程人员能够对不同的函数进行调用,但在C++或Java语言中,都没有提供与之相对应的对象指针。分布式构件采用构件接口引用(Reference)机制来实现这一功能,引用是在功能上与指针类似的机制,它们都用于定位被操作者,但引用可以在其他机制的支持下(例如ORB)跨越计算机、操作系统等不同的运行空间对构件或接口进行精确定位。

构件连接的目的是在构件间建立起调用关系,这就需要将被调用者的引用填入到调用者之中。这一过程建立在事件的基础上,调用者构件状态的变化可以被视为是一种事件(例如属性值的变化等),如果根据设计要求,该事件的发生将对其他构件具有影响,则该事件就必须被传播到相应的构件中,这种事件向外传播的过程即是调用其他构件中操作的过程。实现这一过程的关键之处在于调用者内部包含一个引用队列,用以记录各个被调用者的引用。

而欲监听该事件的构件则主动向事件源—调用者进行注册,将自身的引用加入到队列中,并注明监听何种事件。其结构如图3。当事件发生时,调用者构件即从引用队列中查找对该事件感兴趣的构件,对相应的方法进行回调(Callback)。

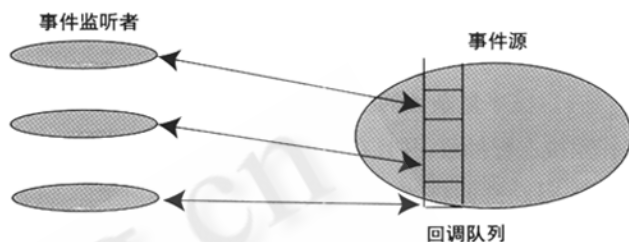


图3 基于事件的构件连接结构

### 2.3.2 构件组装

对构件进行复用的目的在于使用目前可获得的构件构造出可运行的应用程序,或者更大的构件。构件组装过程即是利用现有的构件加入到一个框架中,然后将所加入的构件利用事件进行连接的过程。该过程既可以对应于一个静态的构件设计行为,又可以对应于一个动态的构件配置行为。

在构件之间建立起连接是构件组装的重要过程,其基本原理已经在前面做了简单介绍。在实际的组装过程中,由于构件存在层次结构,使这种连接在跨越不同的层次时变得复杂,需要在包含者导航功能的支持下下进行。另外,事件源与事件监听者对应的接口类型可能存在的不一致,通常可引入一个适配器,对二者进行必要的协调,使通信得以进行。适配器还可以负责区分事件的触发方式,对于向同一个事件注册的监听者,存在多种触发事件的方式:事件源同时触发所有的监听者,事件源也可以一次仅触发某个或某些监听者。

上述构件组装方式的特点在于使用回调队列将不同的构件直接插到一起,所以我们称之为插接式组装方式。除此之外,还存在另外一种粘接式组装方式,它采用脚本语言对不同的构件进行组装。脚本描述能够将不同的构件粘接在一起,它不仅任意调用构件输出的方法,同时还可以被构件产生的事件所触发。由于脚本的编写比较自由,具有很大的灵活性,对于插接式组装方式难以实现或实现现代价较大的情形可以采用粘接式组装方式完成构件间的组装。

### 2.3.3 构件组装工具

构件组装工具是构件技术的重要组成部分,它具有

浏览构件、图形化表示构件以及图形化操作构件等多方面的功能。

(1) 构件浏览。组装工具首先应该能够浏览被复用的构件的信息,包括构件层次、构件属性、构件分布等等。这种信息通常被存放在构件池中,开发人员根据设计需求,在这些构件中查找满足条件的构件。这种需求可能是一个功能需求,即重点在构件的某种操作上;也可能是一个实现结构上的需求,即重点在构件的某种属性上。对于满足条件的构件,可以根据构件分布信息将其实例化,并在组装空间中表示出来。由于构件可能存在依赖关系,因此可以在实例化一个构件之前将其依赖的构件首先实例化。

(2) 构件的图形化表示。尽管分布式构件多数是在运行过程中不可见的构件,在设计过程中为便于对构件进行操作,必须将其在构件组装空间中以图形化的方式表示出来。构件的图形化表示在能力上与构件的文本表示(CDL描述)是等价的,但图形化表示可以根据需要突出表示开发者最关心的信息,而将不重要的信息予以屏蔽,所以有助于降低开发过程的复杂程度。

图形化构件的另一个好处是可以采用“拖”、“放”的形式将构件装入另外一个包含者构件中,并在拖放过程完成的同时完成实例间的包含操作。

(3) 构件的关联。事件关联是构件组装的另外一项核心内容,事件源与事件监听者通过一条有向线进行事件间的关联,事件源对于构件的某一属性,监听者对应于另一构件的某一操作。该动作完成的同时,事件监听者完成了自身向事件源的注册操作。

### 3 结束语

分布式构件是一种特殊的构件,它必须遵从一般的构件模型,但它又在某些方面扩充了构件的内容,象构件间的动态连接等,因此对分布式构件的研究将促进构件技术的研究。另一方面,网络技术的迅速发展推进了对分布式软件的需求,而基于分布式构件的网络软件开发则被认为是一条很有前途的方向。■

#### 参考文献

- 1 The Common Object Request Broker, Architecture and Specification, Revision 2.2, OMG Document Feb. 1998.
- 2 Robert Orfali, dau Harkey, Jeri Edwards, The Essential distributed Objects Survival guide, John Wiley & Sons, Inc.
- 3 CORBA Components, OMG Draft Document, orbos/97-11-24.
- 4 王千祥等, 分布共享对象的语义与实现, 南京大学学报, 1995. 10.