理想的构件模型与性能分析*)

Ideal Component Model and Capacity Analysis

王斌君　王靖亚　郝克刚
(西北大学软件工程研究所　西安710069)

Abstract　After analysing the problem of Object-Oriented technology, advantages and disadvantages of three typical component technology are appraised. According to software architecture philosophy, an ideal component model is given, and the legacy and dynamic maintenance of the component technology is covered.

Keywords　Object-Orientation, Component, Legacy, Maintenance

1　引言

面向对象的软件开发方法是当今软件开发的主流方法。面向对象软件开发方法的优点主要体现在软件的分析、设计和源代码等的开发阶段，而源代码经过编译、连接后得到的可执行软件则像是铁板一块，不可改变和重用。这一问题使得我们原本期望在开发一个项目时也可以象硬件市场上购买硬件一样在软件市场上购买软件配件这一目标无法实现。本文介绍的构件技术是一种软件实现的方法和技术——构件技术，它是面向对象方法在目标代码级的完善和补充。

目前，流行的构件技术有三种，这些构件技术都在一定程度上解决了面向对象方法在目标代码级上的封装、模块化和重用等问题，它们是对面向对象技术的完善和补充。但是，这些构件技术在具体实现细节上都存在有一些不尽如人意的地方，本文讨论了它们各自的特点及缺陷，并给出了一种理想的构件模型。最后，对利用构件技术实现软件的遗产工程和软件的动态维护等问题也进行了探讨。

2　目前流行的三种构件技术的特点及缺陷

目前，在软件市场上有三种具有代表性的构件技术流派，它们分别是 COM[1,2] (Component Object Model，对象构件模型)、JavaBean[3,4] 和 CORBA[5~8] (the Common Object Request Broker Architecture，公共对象请求代理体系结构)，这三种构件技术模型是由不同的机构提出的，各有其优缺点，下面分别加以分析：

(1)COM 的特点分析

COM 是由 Microsoft 公司推出的构件接口标准。COM 构件是一种二进制标准的构件技术，它实现简单、比较实用。在空间方面，接口编译以后生成的二进制代码的结构要满足一定的内存结构，它不依赖于任何语言。正是通过这种统一的二进制代码结构，不同语言之间的互操作和代码的共享才能得以实现。在时间方面，COM 在接口与实现之间应建立一种弱的、松散的耦合，在必要时这种弱的耦合能通过接口找到其相应的实现，并且在不影响所有客户使用的前提下可以将编译好的二进制代码中的一个接口实现换为另一种接口实现，从而将客户和构件本身真正地分开，这有利于以后版本升级和维护。COM 构件的主要特点如下：

COM 是通过 DLL 机制实现对构件生命期的管理的。DLL 是构件的一个实现服务器，当要使用某个构件时，客户通过 Win32的标准函数 HISTANCE LoadLibrary 将实现构件的 DLL 调入内存，然后通过调用标准函数 FARPROC GetProcAddress 得到函数实现的地址，就可以将构件的指针与构件的实现(在刚调入的 DLL 中)联系起来，从而完成指针与实现的松散耦合的要求。另外，DLL 与装载它的 EXE 被分配在同一个进程的地址空间，这一技术减少了实现上的很多麻烦，客户多次装载的某个 DLL 在内存中只保留一个拷贝，这一技术节省了内存空间。

COM 的最初动机是共享人类开发的、丰富的软件资源，DCOM 机制实现了在 Internet 上透明地访问网

*)西北大学青年科研基金项目资助课题。王斌君　副教授，博士研究生，主要研究方向为软件工程、软件理论和计算机安全等。王靖亚　讲师，硕士研究生，主要研究方向为软件工程和计算机网络。郝克刚　教授、博士导师，主要研究领域为软件工程、软件理论。

上的共享资源的方法，它使客户像访问本机构件接口代理）一样透明地使用远程构件。DCOM 更加丰富了 COM 的共享机制。

COM 通过 IunKnown 接口提供了一种与其它构件进行交互操作的标准行为，其中包含的三个函数 QueryInterface（查询接口）、AddRef（增加构件对象引用数目）和 Release（释放）为分别独立创建的客户和构件能够以一种统一的查询方式进行通讯以及节省内存、共享构件提供了强有力的保证。

COM 提供的 Idispatch 接口使得它可以通过一个标准的接口提供服务，而无须知道构件中有无特定的接口。这种功能对某些脚本语言和宏语言非常重要，因为在这些语言中无法展示对象的某个接口的知识。所以，在构件中提供一种通过函数名称执行函数的简单方法非常必要。正是这种简单方便的机制，也被视为 COM 的自动化。

注册表是一张将 GUID、接口易记名字和实现该接口的文件联系起来的表。COM 技术利用这张表，客户可以用简单易记的名字访问接口，而系统内部可查找到其相应的 GUID 以及实现该接口的文件，而且，当接口的实现发生变化或移动时，只需修改该注册表中相应项即可，这一技术使得这一过程对客户是透明的。

由于 COM 是一个二进制标准，而任何语言都要被翻译成二进制机器代码才能最后执行，所以，COM 可以作为各种高级语言的互操作的中间桥梁；按 COM 标准实现的软件也可以被所有的语言环境所共享，这种机制还会给软件开发带来诸多好处。但是，实现这种技术的二进制代码是依赖于机器的，不能满足移动计算的要求。另外，COM 是由 Microsoft 公司推出的构件接口标准，原则上讲这些构件可以建立在任何环境下，但现有的 COM 构件大多都依赖于 Microsoft 环境，在 Unix、Macintosh 等操作系统环境下可复用的构件还很少。因此，很难在其它操作系统环境下实现二进制代码的共享。

（2）JavaBean 的特点分析

JavaBeans 是为了解决平台依赖性而提出来的一种软件构件技术，而这种平台依赖性问题对进一步提高软件复用性是至关重要的。

Java 本身是独立于平台，适宜用于 Internet 环境，易于构造 Browser/Server 结构的软件。但是，Java 是在源代码级的复用技术。JavaBean 是 Javasoft 公司利用 Java 特殊的 ByteCode 机制推出的解决平台依赖性问题的构件接口标准，是在目标代码级的复用技术。JavaBean 提供的标准接口为在目标代码级的动态组装、版本升级、维护提供了保证，一些相应的可视化工具也为方便、有效地定制 JavaBean 和建立应用程序提供了方便。正像 JavaSoft 所描述的，JavaBean 是“一次性编写，在任意地方运行，在任意地方可重用”。JavaBean 有以下主要特征：①非平台依赖性。JavaBean 利用 ByteCode 机制生成可以在任何环境下直接执行的代码，这一点为在当今的 Internet 环境下开发可共享的软件成份和解决移动计算问题提供了独有的、不依赖于平台的软件开发环境。②属性管理。属性的类型分为一般属性、索引属性、依附属性和约束属性，通过标准的命名约定来定义它们相应的访问方法，使 JavaBean API 能用统一的方式对属性进行管理。③内省功能。是一种将构件的内部结构展现给外部的机制。开发者只需要对构件的属性、方法和事件的命名和类型符号遵守一个约定，通过标准的 JavaBean API 就可了解到 Bean 的任何内部信息。

JavaBean 的事件处理模型是基于现存的 AWT 事件处理模型的。它决定 Bean 如何对它自身状态的变化做出反应，以及如何将这些变化传递给应用程序和其它 Bean。JavaBean API 通过将一个事件接收器注册到该事件上完成外部对事件的定制。JavaBean 的其它服务主要是利用 JavaBean 提供的标准接口实现的，这里不再赘述。

JavaBean 存在的问题：JavaBean 主要解决了平台依赖性和可视化构件的一些问题，但它只提供了在 Java 环境下的目标代码共享机制，在不同语言之间的互操作功能还是很弱的，增强 JavaBean 与其它构件技术之间的互操作是 JavaBean 今后发展的一个主要课题。

（3）CORBA 的特点分析

CORBA 是由 OMG（对象管理组）1990年首次提出的，主要为了解决分布式、异质的软件和硬件环境下对象之间的互操作问题，CORBA 具有以下特点：

CORBA 将传统的面向对象模型和分布式问题的对象计算模型有机地结合起来了。对象是分布式计算模型中理想的节点描述模块，既可以是客户，也可以是服务器，这些对象可以自由地分布于计算机网络上。通过 CORBA，对象可网络透明地相互访问，CORBA 屏蔽了位置信息和计算机的软硬件环境，建立了一个统一的分布式软件开发平台。

CORBA 是一种以 IDL 为桥梁、基于 ORB 中间件的构件技术。它以 IDL 为标准，与实现构件接口的语言、软件和硬件平台无关。

CORBA 是一种国际标准，有广阔的发展前景。但 CORBA 目前还存在着一些问题：CORBA 构件的实现是依赖于软、硬件环境的。CORBA 构件一旦实现，尽管它们之间可以实现无缝的互操作、可以被共享和使用，但它们只能在一种环境下运行，无法满足 Internet 网上大量的移动计算的需求。

3 理想的构件模型及软件的体系结构

一个构件不仅给客户提供一个服务接口，而且还可能向别的构件提出接口要求，这种要求也是外界控制构件的一个良好契机。现有软件系统中的回调函数(Callback)、钩子(Hook)函数和各种控件(ActiveX 等)都是这一要求的实例。作为构件应该考虑这种通用的要求，在构件结构中反映这种机制。图1给出满足这种要求的理想构件模型。它由三部分组成：服务接口、构件体(构件的实现)和请求接口。一个理想的构件可以没有请求接口，但不能没有服务接口。没有请求接口的构件就是一个不需要客户对它进行控制、不需要定制的我们现在所指的构件。构件的实现采用类似于 COM 和 JavaBean 的方式，制定一个 ByteCode 级的标准，改造现有的编译系统，使它们所编译的目标代码为 ByteCode，从而完成目标代码级的构件。这种构件与语言无关、与软硬件平台无关，可满足 Internet 上大量的共享和移动计算要求。

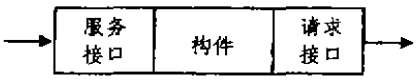

图1 理想的构件模型

现有的构件模型只解决了构件的复用，但未解决由构件形成的应用框架复用问题。而一些典型的应用程序的体系结构是可重用于多个应用和不同环境的，特别是同一应用领域的不同应用之间，这种复用就更加明显和重要。因此，应该有一种既能便于结构复用、又能对结构复用、还便于以后对生成的软件进行维护的方法。

从软件体系结构的角度看，构件技术的构件只是提供一个软件模块的实现，作为一个软件系统，它还需要连接子(如管道、层次、基于事件的隐式调用等等)作为连接软件模块的"粘合剂"。管道和过滤器模型最容易实现软件模块的集成，但它只能处理简单的、流式的应用，没有普遍性。层次模型可提供不同级别的抽象，但层与层之间存在着紧密的耦合，而且这种模型也没有普遍性。基于事件的隐式调用的方式是构件向系统发出请求，已经向系统注册响应该事件的构件就响应该事件。这种模型将调用者和被调用者彻底地分开，这种软件体系结构模型具有很强的灵活性，也具有通用性。

理想的软件体系结构模型提供类似 CORBA ORB 的"软件总线"，连接到软件总线上的理想的构件应该如图1所示，它只接收事件和发出事件。

传统的基于事件的隐式调用模型中，事件的发送者和接收者并不需要预先知道，接受者是以事件类型区分和响应调用的，但不区分调用；调用者只发出事件，但不指定事件的接收者，正是这种机制使基于事件模型具有区别于别的模型的可扩展性优点，以及灵活、方便地组织和集成构件的能力。但是，这种集成构件的方式不能处理相同构件类型的不同实例的事件发送者和接收者之间的交叉引用问题。

在理想的软件体系结构模型中，对基于事件的模型作了如下改进：使发送者发出的消息为如下的结构 t理想模型)：

(发送者，事件类型，事件参数)

接收者根据需要区分或不区分"发送者"。

而传统的软件系统中模块之间的调用关系可抽象为如下的结构(调用模型)：

(接收者，事件类型，事件参数)

发送者模块明确指明接收者，以及要求接收者响应的事件(函数、过程或方法)。

两种构件模型中访问方法的区别是：前者由接收者选择发送者，而后者则是由发送选择接收者。其实，传统的基于事件的软件系统中，事件的发出者和事件的响应者之间消息的格式可表示为(事件模型)：

(事件类型，事件参数)

它是一种介于前两者之间的模型。

这三种模型各有优缺点：调用模型是事件的发出者选择响应者，系统实现容易、效率比较高，但系统一旦建造就很难修改；事件模型是响应者选择调用者，软件系统的体系结构非常灵活，模块容易集成和动态组装，但响应者不能区分不同的调用者。

理想模型则提供了更大的灵活性，让响应者根据需要选择调用者或者忽略调用者。它具有如下的优点：软件易扩展。新增加的构件只要识别已有构件发出的事件即可，而传统的构件模型中需要已有构件制定新增加的构件，这就需要修改原有构件。可进行广播式(broadcast)调用。使理想模型中的发送者为空，它表明任何能响应该事件的构件都可响应该事件，从而完成广播式调用。可进行组播式(multicast)调用。使理想模型中的发送者为一个组名，它表明任何能响应该事件的同一个组的构件都可响应该事件，从而完成组播式调用。

为了使这种模型具有资源共享、开放、网络透明和异质软硬件环境的互操作，该模型可在 CORBA 的事件服务之上实现。

4 构件模型的可动态维护性

构件技术的初衷是为了能够充分地利用和共享在各种环境下、用各种高级程序设计语言开发的软件产

品，它是一种目标代码级的软件复用技术。正像我们在前面分析的那样，通过接口这种不依赖于具体语言的中性机制，使各种语言之间可以互操作，也就是说一种语言可以通过接口无逢地访问另一种语言开发的软件，而不需要移植工作，这大大地提高了软件的复用程度。

但构件为我们提供的好处不仅在于此，通过它还能为我们提供解决遗产工程的问题。所谓遗产工程是指大量的、已存在的软件，遗产软件是前辈们花费很多时间、精力和资金开发出来的，是智慧的结晶和劳动的果实，在采用更好的新技术时，不要将以前开发的软件简单地抛弃掉，特别是那些在某个领域中广泛使用的、运行良好的通用软件，应该通过一个良好的机制对其进行改造利用，这不仅可以减少重复性的劳动、压缩软件开发周期，而且可以节省巨额资金。

我们可以对遗产软件的每个成份用构件技术开发一个接口，即用接口对原软件成份进行包装，通过这种方法，可以复用大量的原系统的实现，从而完成对遗产软件的改造。改造后的软件成分还可以通过接口在非原系统语言环境下被使用，大大拓展了复用的力度。

下面我们着重讨论构件为我们提供的另一个不为人注意的特性——可动态维护性。

众所周知，软件的维护性是很重要的，而如何在目标代码级实现维护还是鲜为人知的，我们曾经为中原油田开发的项目 SIDIS 系统就有这种要求[10]。存在的问题是：当我们在原系统上扩展一个功能后，却很难将完善的软件加入到原来的 SIDIS 系统中，其原因主要是可执行的目标系统是一个铁板一块状的结构、无法对其进行修改、替换和增加。但是，对目标代码的维护要求却是很普遍的一个问题。

下面我们给出一个利用 COM 和 CORBA 构件技术实现对目标代码进行维护的方法。无论是 COM 还是 CORBA 构件技术，都是通过对软件部件的接口和实现的分离，构造不依赖于语言环境的、可复用的目标代码的。这些软件部件在系统集成后，也是相对独立的，它们都是通过接口相互联系。也就是说，组成系统的构件在目标代码的可执行软件系统中保持其相对独立性，这就打破了传统软件的目标系统铁板一块的局面，为软件在维护阶段对目标代码中的"部件"进行替代成为可能。

在 COM 中，构件的实现都存在于相对独立的 .exe 或 .dll 文件中，只有在系统执行时才把它们调入内存，通过接口完成调用者和被调用者之间的联系。如果系统在维护期间需要修改某个构件，只需修改相应构件的源代码（或重写），然后编译源代码，使其生成的文件代替原有的 .exe 或 .dll 文件，即可达到维护目的。这种维护，不需要通过开发商、可由最终用户直接进行，我们可以称其为动态维护[10]。实际上，构件版本的升级就是通过这种手段进行的。

对于 CORBA 来讲，也是采用类似的方式，CORBA 中的 CORBA_BOA_change_implementation 可以完成接口与实现的静态相关性。而 CORBA 的动态调用则直接支持这种功能。

以上是对已有构件的修改和完善。那么，怎样动态地给系统增加新功能呢？通过以下两种方法都可以为系统增加新的功能：修改请求构件，使其发出请求；增加新的构件，完成新的功能。在增加系统功能的过程中，系统中其它成分无需变动。

总结 面向对象是一种非常好的软件开发方法，其优越的分析方式能直观、有效地把握问题的本质；其良好的设计技术能直接映射问题空间，能建立一个具有可复用部件的、有良好体系结构的软件；但面向对象的实现技术使得面向对象的复用、可维护和灵活定制等优势大打折扣。一个设计良好的面向对象结果，并没有要求某个对象要用某种语言、在某种环境下实现，并没有要求这些对象一定要绑定在同一个进程之中、同一台计算机或在不同的计算机上，而这些问题用面向对象语言实现时都不得不认真加以考虑。

构件技术正是在这种环境下提出的，它是对面向对象技术的完善和补充，它是目前最重要的软件开发技术。这种在目标代码级的接口标准，使得软件可在目标代码级重用大量已有的软件实现、可动态地组装一个软件，提供了各种软件模块之间的可操作性，这为最终用户定制一个软件、以及软件本身的进化和维护提供了很好的机制。通过接口可以包装一个已有的软件系统，也为遗产软件的再利用提供了技术保证。

参考文献

1 [美]Rogerson D. COM 技术内幕 杨秀章译. 北京：清华大学出版社，1999
2 [美]Kirtland M. 基于组件的应用程序设计. 北京：北京大学出版社，1999
3 Coulouris G, Dollimore J, Kindberg T. Distributed Systems Concepts and Design, Second Edition. Addison-Wesley, 1996
4 [美]Morrison M. JavaBeans 使用手册. 周苏明，常征等译. 北京：机械工业出版社，1997
5 OMA. The Common Object Request Broker: Architecture and Specification 2.2 1998
6 OMA. CORBA Service: Common Object Service Specification 1997
7 Natan R B. CORBA A Guide to the Common Object Request Broker Architecture. McGraw Hill, 1995
8 [美]Orfali R, Harkey D, Edwards J. 智能 CORBA. 陈章渊，张学东，于秀山等译. 北京：电子工业出版社，1999
9 Shaw M, Garlan D. Software Architecture-Perspectives on an Emerging Discipling. Prentice Hall, 1996
10 王斌君，等. 可动态维护的软件的体系结构. 西北大学学报，1998，28(3)：193～197