CAR包容技术的设计与实现

陈洪光[1],陈 榕[2]
(同济大学 基础软件工程中心,上海 200092)

摘 要:传统构件技术的包容都是静态的,这显然不利于使用,而设计一种动态包容技术将大大增加构件的灵活性。基于CAR构件设计并实现了一种动态包容技术,此技术充分利用了CAR的基于二进制的AOP编程模型,将包容器设计成方面构件类AObjectContainer,从而使聚合了AObjectContainer的普通构件类轻松地具有了包容能力。此方法具有使用简单,可动态控制被包容对象等特点。

关键词:CAR;包容;方面;AObjectContainer;聚合;动态
中图分类号:TP311.52 **文献标识码**:A **文章编号**:1673-629X(2008)12-0015-04

Design and Implementation of CAR Container

CHEN Hong-guang[1], CHEN Rong[2]
(System Software Engineering Centre of Tongji University, Shanghai 200092, China)

Abstract: Because the containers of common component is static, use them inconveniently. So in this paper, design and implement a container based on CAR. This technique uses CAR AOP model based on binary system adequately. It designs container to be an aspect component class: AObjectContainer. So a normal component class will simply have the ability of containing if it has aggregated AObjectContainer. This technique has many characteristics such as simply using and dynamically controlling class being contained.

Key words: CAR; container; aspect; AObjectContainer; aggregate; dynamic

0 引 言

CAR(Component Assembly Runtime)是具有自主知识产权的先进的构件技术,已在Elastos操作系统、Windows 2000等系统上实现。CAR构件技术定义了一套面向构件的编程规范,使得二进制构件能够自描述,能够动态链接,实现了目标代码级别的软件复用[1]。

与传统软件需要升级一样,CAR构件也不例外。当已经有一个构件之后,同样希望对其进行加工、扩展以满足自己的需要,并可能会希望用改造后的构件替代原有构件。在C++中,对类的改造是通过包容和继承实现的。与C++相对应,CAR构件技术同样可以通过包容和聚合来实现对构件的改造[1]。

文中设计并实现了一种全新的CAR包容技术,它通过包容器方面构件类AObjectContainer来对被包容在一个构件对象里的其它构件对象进行管理。这样就可以把程序员从复杂但相对次要的包容细节中解脱出来,从而把主要精力放在构件的逻辑细节上。AObjectContainer是一个特殊的方面构件类,它作为一个外部构件可以包容若干个同类或不同类的其他构件,这些其他构件作为内部构件被包容在AObjectContainer构件类之中。此时,AObjectContainer构件仅仅是内部构件的一个客户而已,它可以通过对象枚举器来枚举内部构件对象,并使用它们。

1 CAR构件技术

CAR构件技术是在总结面向对象以及面向构件编程模型的历史和经验基础上,为更好地支持以Web Service为代表的下一代网络应用而设计的构件化编程模型。为了在资源有限的嵌入式系统中实现面向中间件编程[2],又能得到C/C++的运行效率,CAR没有使用基于中间代码的虚拟机机制,而是采用了C++编程,用Elastos SDK直接生成运行于"CAR构件运行平台"的二进制代码的机制[2]。利用在不同操作系统上实现的CAR构件运行平台,可以使CAR构件的

收稿日期:2008-03-27
基金项目:国家863计划资助项目(2001AA113400);国家移动通信产品研究开发专项项目(财政部(财建[2005]182号),信息产业部(信部请函[2005]297号))
作者简介:陈洪光(1982-),男,河北唐山人,硕士研究生,研究方向为嵌入式操作系统、系统软件支撑技术;陈 榕,博士生导师,教授,研究方向为嵌入式系统、构件技术。

二进制代码实现跨平台兼容[1,3]。

可以说,CAR 为构件软件和应用程序之间进行通信提供了统一的标准,它为构件程序提供了一个面向对象的活动环境。CAR 标准包括规范和实现两大部分,规范部分定义了构件和构件之间的通信机制,这些规范不依赖于任何特定的语言和操作系统,只要按照该规范,任何语言都可使用;CAR 标准的实现部分可能是 Elastos 操作系统,Elastos 操作系统为 CAR 规范的实现提供了一些核心服务。同样,CAR 标准的实现部分也可能是 Elastos 虚拟机[1,3]。同时,CAR 提供了一种基于二进制的 AOP(Aspect Oriented Programming,面向方面编程)实现,能够灵活地实现构件级别的动态插入、拦截,从而使构件具有动态组合、扩展的功能。CAR 的 AOP 机制使用户能够在不修改源代码的情况下,方便地将一个普通 CAR 构件类与若干 CAR aspect(方面)构件类动态聚合起来,从而生成一个具有多个 CAR 构件类的所有接口的新构件类。与动态聚合对应,aggregate 属性用于修饰普通构件类,能够将 aspect 与所要定义的普通构件类结合从而实现静态聚合[1]。

2 CAR 包容技术

2.1 CAR 包容技术简介

一个完善的构件技术必须实现包容,CAR 也不例外。而通过对象包容器方面构件类 AObjectContainer 来管理被包容对象的方案,能够很好地把构件编写者从包容的实现细节中解脱出来。AObjectContainer 作为 CAR 的一个特殊的方面构件可以包容若干个同类或不同类的其他构件,这些构件作为内部构件被包容在 AObjectContainer 构件中。假如一个普通构件聚合了 AObjectContainer,那么它就可以通过 AObjectContainer 包容其它的构件对象。

在 CAR 技术中,可以把这个 AObjectContainer 看成是一个箱子或者是一个口袋,它可以将多个同类的或不同类的构件对象打包起来。AObjectContainer 这个方面构件本身不是主体的属性或谓词,但它可以与某个主体的构件对象聚合,这样该主体构件对象就可以调用 AObjectContainer 内部所包含的构件对象的接口和方法,从而使主体构件包容了 AObjectContainer 的内部构件。这里 AObjectContainer 提供了一种中介,通过它外部构件才包容了内部构件。而且接下来可以看到,AObjectContainer 包容的内部构件是可以添加或删除的,所以这种包容又是动态的。

AObjectContainer 定义了 IObjectContainer 接口,IObjectContainer 接口包含了一组通用方法,这些方法为包容提供了支持。这些通用方法如下:

```
Add(IObject * pObject); //向包容器中添加构件对象
Remove(IObject * pObject); //从包容器中移除构件对象
GetEnumerator(IObjectEnumerator * * ppEnumerator); //获得被包容构件对象的枚举器
GetCount(Int32 * pCount); //获得被包容构件对象的个数
Dispose(); //释放包容器,并释放资源
```

IObjectContainer 获得的枚举器接口 IObjectEnumerator 主要用于内部构件对象的枚举。IObjectEnumerator 接口提供了以下方法:

```
Current(PObject * ppObject);
//获取枚举器中的当前对象
MoveNext(Boolean * pSucceeded); //设置下一个对象为当前对象
Reset(); //重置枚举器为初始状态
```

CAR 包容与 C++ 包容类似,但与 CAR 的其他技术一样,CAR 包容也是从接口一级实现的,外部构件包容指向内部构件接口的指针。此时,外部构件只是内部构件的一个客户而已,它会将内部构件的接口当作自己的接口使用,如图 1 所示[4,5]。

外部构件也可以通过将调用转发给内部构件的方法来重新实现内部构件的接口,并且外部构件也可以通过在内部构件接口之前或之后加入一些代码的方法改造内部构件,如图 2 所示[4,5]。

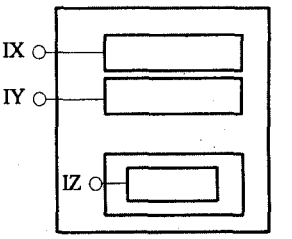

图 1 使用内部构件 IZ 接口的外部构件

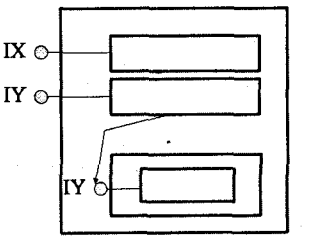

图 2 复用了内部构件接口 IY 的外部构件

2.2 CAR 包容技术的设计目的

CAR 构件具有包容能力的一个原因来源于对传统面向对象编程模型不足的改造。在传统的面向对象编程模型中,继承主要指的是“实现继承”[6],这种继承主要是指派生类继承其基类的代码或实现。CAR 构件技术不支持“实现继承”,取而代之,CAR 支持“接口继承”。这种继承指的是派生类继承其基类的类型或接口[1]。

CAR 不支持"实现继承"的主要原因在于这种继承方式使得一个对象的实现与另一个对象的实现紧密关联起来[4,5]。在这种情况下,基类的实现改变后派生类必须也要做相应改变,否则派生类将无法正常运行。在中等规模的 C++ 程序中,这或许并不成为问题,因为在这种情况下,一般能够获得相应源代码而进行修改。但对大型的 C++ 程序,这种修改无疑会浪费过多的时间,而且在相当多的情况下,是无法直接获得源代码的。这也正是为什么一些用 C++ 编写大型软件的专家们强烈建议基于抽象基类来构建应用程序的原因[1,6]。

抽象基类是一种纯粹的接口继承,并且也被用于实现 CAR 接口。由于任何人可以用任何语言在任何地点编写 CAR 构件,因此必须有一种严格的方法来保证构件的使用者不会因为构件的改变而受到影响。显然,实现继承无法给用户提供这种保护。

因此,为了保证构件的修改不会影响应用程序的正常运行,CAR 构件技术并不支持实现继承。这样,为了保证 CAR 的功能不会因为不支持实现继承而有所损失,必须设计一种实现继承的替代技术。在 CAR 构件中,包容完全可以模拟实现继承。当然实现继承比包容更方便一些,但为了系统的强壮性而做出这种选择是恰当的。

CAR 构件必须具有包容能力的另一个原因来源于 CAR 本身。因为在 CAR 规范中规定:只有 aspect 构件类才可以被聚合[1]。如前所述,聚合是实现构件改造、复用的重要方法,但只有 aspect 构件类才可以被聚合的限制无疑带来了这样的问题:怎么复用一个普通构件类呢? CAR 包容技术恰好可以弥补因这个限制而产生的问题,它可以把一个不可以被聚合的非 aspect 构件类暴露成另一个构件类的一部分。因此,可以说,CAR 的聚合和包容互为补充,很好地支持了 CAR 的构件级别的代码复用,支持了"接口继承"。

2.3　CAR 包容技术的实现

如前所述,通过对象包容器方面构件类 AObjectContainer 来实现 CAR 包容技术是相对简单的。AObjectContainer 的主要逻辑可以用如下的伪代码加以说明:

```
class AObjectContainer : public AObjectContainer
{
    public:
    CARAPI Add (/* [in] */ IObject * pObject); //向包容器中添加构件对象
    CARAPI Remove (/* [in] */ IObject * pObject); //从包容器中移除构件对象
    CARAPI GetEnumerator( /* [out] */ IObjectEnumerator * * ppEnumerator);
    //获得被包容构件对象的枚举器
    CARAPI GetCount (/* [out] */ Int32 * pCount); //获得被包容构件对象的个数
    CARAPI Dispose (); //释放包容器,并释放资源
    CARAPI constructor (); //构造并初始化包容器
    AObjectContainer (); //析构函数
    private:
    int m_nCount; //计数器,用于记录包容的内部构件个数
    int m_nState; //状态,主要用于枚举器
    ObjectContainer m_container; //双向链表结构,用于实际包容内部构件
    CriticalSection m_lock; //互斥锁,用于对 AObjectContainer 的互斥访问
};
ECode AObjectContainer::Add(/* [in] */ IObject * pObject)
{
    ……
    检查互斥锁 m_lock, 实现对 AObjectContainer 的互斥访问;
    ec = m_container.Add(pObject);
    if (NOERROR == ec) {
    m_nState++;
    m_nCount++;
    ……
    }
    离开互斥段, 关闭互斥锁 m_lock, 结束本次互斥操作;
    ……
}
ECode AObjectContainer::Remove(/* [in] */ IObject * pObject)
{
    ……
    检查互斥锁 m_lock, 实现对 AObjectContainer 的互斥访问;
    ec = m_container.Remove(pObject);
    if (NOERROR == ec) {
      m_nState++;
      m_nCount--;
      ……
    }
    离开互斥段, 关闭互斥锁 m_lock, 结束本次互斥操作;
    ……
}
ECode AObjectContainer:: GetEnumerator (/* [out] */ IObjectEnumerator * * ppEnumerator)
{
    ……
    pEnum = new_CObjectEnumerator((_IObject *)this,(ObjectNode *)&m_container.m_head,
    &m_lock, &m_nState);
    * ppEnumerator = (IObjectEnumerator *)pEnum;
```

```
    ……
}
```

上面的伪代码清单中已经包含了 CAR 实现包容技术主要的声明及实现。实际上，CAR 包容器 AObjectContainer 主要是通过双向链表 m_container 实现的，m_container 实际包容了内部对象的指针。通过操作此链表并利用枚举器就可以获得内部对象的接口指针，进而通过此指针使用内部对象。这样，外部对象暴露了内部对象的接口，用户就可以透明地调用内部对象的接口，而且这个调用过程完全由 AObjectContainer 来保证，不需要用户的干预。

通过分析上面的伪代码可以发现，CAR 包容技术实际上是利用了 AOP 技术，设计了一个包容器方面构件类 AObjectContainer。这样，通过在一个普通的构件类上聚合 AObjectContainer 的方法就可以使这个普通构件类具备包容能力，而 AObjectContainer 中保存的对象指针就是内部对象的一个枚举。此时，外部对象就可以通过对象枚举器来枚举内部构件对象，进而使用内部对象提供的接口。

2.4 CAR 包容技术的特点

从上面的例子可以看出：CAR 通过 AObjectContainer 实现的包容技术是可以由用户动态控制的，用户完全可以动态地管理构件对象之间的包容关系、内部对象的生命周期等，而其他构件包容技术往往是不具备这种动态性的。

以 COM 为例，假设希望对象 B 能够包容对象 A，其 C++ 伪代码往往如下：

```
//ISomeInterface 记录了对象 A 的接口指针，IOtherInterface 记录了另一个对象 D 的接口指针
Class CB:public ISomeInterface, public IOtherInterface
{
    ……
    private:
    ISomeInterface * m_pSomeInterface;
    ……
}
HRESULT CB::Init()
{
    ……
    HRESULT result = ::CoCreateInstance(CLSID_ComponentA, NULL, CLSCTX_INPROC_SERVER,
    IID_ISomeInterface, (void * *)&m_pSome Interface);
    ……
}
HRESULT CBFactory:: CreateInstance ( IUnknown * pUnknowOuter, const IID &iid, void * * ppv)
{
    ……
    pObj = new CB();
    pObj->Init();
    ……
}
```

分析上面的伪代码可以发现：COM 的包容技术是不具有动态性的。这是因为对于 COM 构件，构件间的包容关系是在编写外部构件的时候已经确定的。同样，因为在编写外部构件时要确定内部构件，这样内部构件的生命周期也就不具有动态性了。而且更为重要的是，以 COM 为代表的 Factory 模式（即由 Factory 来创建接口）并没有解决好代码的耦合问题，显然，要使用 IOtherInterface 就需要修改代码了[4,5]。

综上所述，CAR 包容技术的最大特点就是动态性，用户完全可以动态地管理构件对象之间的包容关系、内部对象的生命周期等，这大大增加了 CAR 构件模型的灵活性。

3 结束语

利用 AOP 中的 aspect 思想，结合 CAR 构件技术实现了一种全新的包容技术，进而实现了 CAR 构件的“接口继承”。

该方法充分利用了 AOP 技术的分离关注点的方法，将包容器设计成一个方面构件类 AObjectContainer，当某个构件对象需要具备包容能力时，只需要聚合此方面构件类就可以轻松地具有包容能力，并且被包容对象是可以动态添加、移除、使用的，从而改进了 COM 等构件模型的包容技术。

参考文献：

[1] Koretide. CAR's Manual[M/CD]. 2005-06. http://www.koretide.com.cn, 2004/2005.6.

[2] Chen Rong. The Application of Middleware Technology in Embedded OS[R]. Hangzhou: Workshop on Embedded System, In Conjunction with the ICYCS(6th),2001:1-3.

[3] Koretide. Elastos 2.0 Operating System Manual[M/CD]. 2005-06. http://www.koretide.com.cn, 2004/2005.6.

[4] 潘爱民. COM 原理与应用[M]. 北京：清华大学出版社，1999:25-32.

[5] Rogerson D. Inside COM[M]. Washington, USA: Microsoft Press, 1997.

[6] 朱其亮，郑 斌. CORBA 原理及应用[M]. 北京：北京邮电大学出版社，2001.