CAR 构件编程技术中的自描述特性[1]

郑 炜[2]，陈 榕，苏翼鹏，殷人昆
（清华大学深圳研究生院）

[摘要] 从面向对象编程到面向构件、中间件编程，是网络时代编程技术的飞跃。构件与接口数据的自描述，为从二进制级上把接口与实现分离提供了方便，并达到接口可以跨地址空间（或者说可以远程化）的目的。构件及接口数据的自描述是 COM 的理论基础及立足点之一。CAR 构件编程技术深入贯彻了这一思想。论文研究了其相关的设计与实现，并探讨其对网络时代编程模型的参考意义。

[关键词] CAR 中间件 自描述

CAR Component Oriented Programming and its self-description property

Zheng Wei，Chen Rong, Yin Renkun
(Tsinghua Graduate School at ShenZhen)

[Abstract] From object oriented programming to component/middleware oriented programming is a great evolution in Internet era. The self description of components and interfaces which is regarded as one of the foundation for COM technology provide a mechanism for separating interfaces from their implementations, by that it can make interfaces across local address space (or said across remote machines). This idea is carried through in CAR Technique. This paper introduces CAR component programming technology and its design and implement which concerning its self-description property, and discusses its significance on the programming model in Internet era.

[key word] CAR, middleware, self-description

1、引言

网络技术普及的初期，以传统的 TCP/IP 系统为主，与之相适应的是客户端/服务器（Client/Server）的"两层结构"计算模型。随着网络应用的深入，人们对与之相适应的软件编程模型的认识更加深刻，从而出现了"三层结构"计算模型：客户/中间件/服务器（Client/Middleware/Server）。这种三层结构模型中的重要概念是构件和中间件。三层结构模型适应了因特网时代软件与软件之间信息交互的需要，这是因特网环境下，软件之间协同工作的一种新的运算模式。

软件间的信息交互对编程技术提出了新的要求。在面向对象编程中，对象的概念拘泥于数据结构加运算。对象实现了程序模块的封装，带来了更加清晰的程序结构。但是对象模型毕竟是静态的，不能在运行时动态地组织程序，因而不能适应软件之间信息交换的需要。作为面向对象程序设计语言的 C++，其编译生成的目标代码 obj 文件，并不含有类信息，因此系统不可能利用其提供动态服务。许多人称 Java 语言的出现具有"革命性"的意义，其"革命性"不在于简单的语法、垃圾回收机制、或者跨平台特性，而在于 Java 语言在最后编译生成的.class 文件中包含有类信息。这里所指的类信息，就是一种元数据（Metadata）。元数据通常称为"描述数据的数据"，广义上说，元数据也可以是对运算的描述。元数据结合对象所组成的程序模块就是构件。在客户端/中间件/服务端三层结构中，当客户程序需要调用一个软件服务时，系统负责找到服务程序。当服务程序在进程外或是远程时，只要根据元数据的描述，就可以动态生成代理构件，实现客户与服务连接的环境。这个由系统动态生成的代理构件就是中间件。在这种机制中，客户与服务不再直接接触，所有的通信机制都通过中间件自动实现。无论是利用网络连接地理上分布的各类计算设备为用户提供相对透明的虚拟的高性能计算环境的网格计算，还是为移动中的个体进行自动定位并提供与环境相应服务的普适计算，都需要系统自动提供相应的通信与计算环境，其关键就是中间件技术。从面向对象编程到面向构件、中间件编程，是网络时代编程技术的飞跃。

科泰世纪科技有限公司推出的"和欣"操作系统，已

1. 本文的工作得到国家高技术研究发展计划（863 计划）经费资助（课题编号：2001AA113400，2003AA1Z2090），特此致谢。

2. 作者简介：郑炜，在读硕士研究生，清华大学深圳研究生院。

被列入国家863"十五"计划中的基础软件平台研发重点项目。"和欣"创新性地实现了与微软 COM 构件技术兼容的"CAR 构件技术"。CAR（Component Application Runtime）是一个面向构件的编程模型，也可以说是一种编程思想，它表现为一组编程规范，包括构件、类、对象、接口等定义与访问构件对象的规定。CAR 技术兼容微软 COM 构件技术，但是和微软 COM 相比，它删除了微软 COM 中过时的约定，禁止用户定义 COM 的非自描述接口；完备了构件及其接口的自描述功能，实现了对 COM 的扩展；对 COM 的用户界面进行了简化包装，使得高深难懂的构件编程技术很容易被 C/C++ 程序员理解并掌握。

本文在第2节说明微软COM构件技术在自描述特性方面的缺陷；第 3 节探讨了 CAR 构件技术相对于微软 COM 构件技术在自描述特性上做出的扩展，及其在 CAR 构件编程技术的设计与实现中的具体体现以及实际应用；第 4 节对其未来的发展提出展望。

2、微软 COM 构件技术在自描述特性方面的缺陷

在面向构件、中间件编程模型下，构件可以在本机，或者不在本机而通过网络获得，系统要动态生成中间件，建立用户与构件的联系管道，所依据的是构件中的自描述信息。如果把构件看作软件工厂中封装完毕的"零件"，自描述信息就相当于"规格说明"。因特网时代编程技术充分强调的"构件与接口数据的自描述"，正是 COM（Component Object Model）技术的理论基础及立足点之一。但微软 COM 构件技术，在具体设计和实现上，并没有完全的贯彻这一思想，其不足之处主要体现在以下几点：

1) 构件自描述内容未包含本身相关的运行信息。在微软 COM 构件技术中，构件的一些相关运行信息都存放在系统的全局数据库--注册表中，构件在能够正确运行之前，必须进行注册。构件的自描述内容包含本身运行信息，是构件动态运行于不同的环境下的前提。

2) 导出接口的描述信息依赖其他程序。微软 COM 构件技术对构件导出接口的描述方法之一是使用类型库（Type Library）元数据。类型库本身虽然是与构件的 DLL（Dynamic Link Library）文件打包在一起的，但类型库信息却需要靠系统程序 OLE32.DLL 来提取和解释，而不是由构件自身完成。

3) 缺少对构件依赖关系的自描述。微软 COM 构件技术中，构件只有关于自身接口（或者说功能）的自描述，而缺少对构件依赖关系的自描述。事实上，大多数情况下，一个构件会使用到另一些构件的某种功能，也就是说构件之间存在相互的依存关系。在网络计算时代的今天，正确的构件依赖关系是构件滚动运行、动态升级的基础。缺少对依赖关系的自描述，显然满足不了构件滚动运行、动态升级的需要。

4) 构件程序没有关于自身导入信息的描述。在微软 COM 构件技术中，构件程序没有关于自身的导入信息的描述，构件依赖关系是建立在全局的注册表之上，而注册表信息的增加和修改是通过构件的安装程序来完成。这种做法有以下弊病：

- 所有构件的运行都依赖于注册表，系统在能够运行前必须进行完成的安装，并且经过多个软件的反复安装、卸载后，系统注册表会变得很庞大并且难以维护；
- 由于注册表是一个全局数据库，访问权限不容易控制，注册表容易成为病毒、黑客软件的入侵点；
- 构件间的依赖关系建立在完整安装的基础上，如：A 构件依赖于 B，当 B 没有安装到系统中时，由于系统没有任何关于 B 构件的信息，这时 A 构件也就不能正确运行了。

3、CAR 构件技术在自描述特性上的扩展

3.1 CAR 构件定义语言

CAR 构件定义语言是在 COM IDL 的基础上发展起来。IDL 最初由 OSF 为 DEC 中的远程过程调用（RPC）而定义，后来微软公司在这个 IDL 的基础上扩充，制定了 COM IDL。"和欣"在微软的 COM IDL 的基础上再进行改造，摒弃了微软 COM IDL 中一些不合理的设计，并对一些设计，结合 CAR 编程模型的需要进行了扩展。

CAR 构件定义语言与 COM IDL 的区别主要体现在以下几个方面：

1. 不支持 Library 关键字，以 component 取代。对于 COM IDL，如果 Library 表示的库信息组出现在 IDL 文件中，MIDL 编译器就会生成一个类型库。类型库是一个二进制文件，它包含了 IDL 中描述的标准类型信息。前文中讨论到，类型库信息需要系统程序 OLE32.DLL 来提取和解释，违背了自描述思想。CAR 技术中摒弃了这种做法。一个 CAR 文件仅描述一个 CAR 构件（component），构件的定义由对接口的描述和对类的描述两个部分组成，一个 CAR 文件的内容就由"构件属性"和"构件定义"这两部分的代码组成，利用 CAR 文件编译器编译 CAR 文件所得到的类信息（ClassInfo），将允许 CAR 构件不依赖于其他系统程序即完成自描述，具体在下文 3.3 节中讨论。

2. 不支持 coclass 关键字，以 class 取代。在 COM IDL 中，coclass 语句可以出现在库块的内部，也可以出现在库块的外部。当它出现在库块的内部，

或者在库块内部被引用时，那么 coclass 的类型信息将出现在生成的类型库中。在 CAR 文件中，类的定义非常重要。构件对象是某个类的一个实例，所以必须要有类定义的存在，否则不可能有构件对象的存在。

3. 引入类别 category 关键字。COM 规范声称，把虚接口抽象出来，就是实现了二进制多态。但实际上，把接口抽象出来只是实现了构件方法调用的多态性，并未实现构件创建的多态性。一个构件的使用总要经过创建、调用、消亡这三个过程，只有实现了构件对象创建的多态性才算是完整实现了构件使用的多态。出于以上原因，CAR 构件技术中引入了构件类别（category）的概念。一个构件类别包含了一族公用接口，某个构件类要属于这个类别就必须继承这个类别并实现这个类别包含的所有接口。构件类别本身没有实现代码。

在开发一个 CAR 构件时，首先就是写一个 CAR 文件，然后编译该文件，自动生成工具就可以自动生成程序框架，此后需要做的就是添加处理自己事务逻辑的代码。

3.2 对微软 COM 自描述特性扩展的基本思想

围绕着构件的自描述封装和运行，CAR 技术采用了如下的措施对微软 COM 进行改进和扩展：

1. 在 CAR 编程模型中，把类信息（ClassInfo）作为描述构件的元数据。类信息由 CAR 文件编译而来，是 CAR 文件的二进制表述。

2. CAR 构件技术对构件的依赖关系进行了封装。即把一个构件对其它构件的依赖关系也作为构件的元数据封装在构件中，这种元数据被称为构件的导入信息（ImportInfo）。

3. CAR 构件通过对 ClassInfo 和 ImportInfo 的封装，可以实现构件的无注册运行。并可以支持构件的动态升级和自滚动运行。

3.3 构件的类信息（ClassInfo）

CAR 构件技术中自行实现定义的构件的 ClassInfo 是 CAR 文件的编译结果，也就是 CAR 文件的符号化表示。ClassInfo 被作为构件程序的元数据信息，用于描述构件导出的接口及方法列表。同时 ClassInfo 也是自动生成构件源程序的基础。在编译构件程序时，ClassInfo 会作为资源数据被打包到构件 DLL 文件的资源段中。

在根据 ClassInfo 自动生成的 C/C++构件源程序中，包括了标准构件导出函数 DllGetClassObject 的实现代码，DllGetClassObject 函数完成了标准的 COM 语义：取构件类对象。在 CAR 构件技术对 DllGetClassObject 函数的自动实现中，支持使用一个特殊的 CLSID－－CLSID_ClassInfo 来获取构件的元数据，而不是类对象。自动生成的 DllGetClassObject 的实现代码示例如下：

```
STDAPI DllGetClassObject(REFCLSID clsid,
                         REFIID riid, void** ppv)
{
    if (clsid == CLSID_ClassInfo) {
        // 取构件类信息，pCIClassInfo_指向构件
        // 类信息的起始地址，参数 riid 被忽略，返
        // 回的地址存放在 ppv 中
        *ppv = (void *)pCIClassInfo_;
        return S_OK;
    }
    if (clsid == (REFCLSID)CLSID_CSample) {
        // 取构件 CSample 的类对象
        return _g_CSampleCF.QueryInterface(
                                    riid, ppv);
    }
    ……
    return CLASS_E_CLASSNOTAVAILABLE;
}
```

如示例：当输入参数指定的 CLSID 为 CLSID_ClassInfo 时，DllGetClassObject 返回的不再是某个类的类对象，而是构件的类信息的起始地址。这样，当系统或构件客户需要使用 ClassInfo 元数据时，就可以用 CLSID_ClassInfo 作为输入参数，调用构件的 DllGetClassObject 来取得。以上过程如图 1 所示：

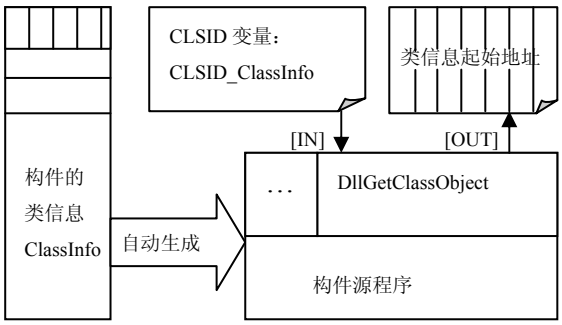

图 1 获取 ClassInfo 的过程

获取构件元数据的示例代码如下：

```
IModuleRef iModRef;HRESULT (__stdcall *
pDllGetClassObject)(REFCLSID, REFIID, void**);
    ……
// 装载构件 DLL 模块
hr = iProcessRef.LoadModule(ezsDllName, NULL,
&iModRef);
```

```
if (FAILED(hr)) {   return hr;}
// 取 DllGetClassObject 函数的地址
hr = iModRef.GetProcAddress(
      EZCSTR("DllGetClassObject"), 0,
      (ADDRESS)pDllGetClassObject);
//调用构件 DllGetClassObject 方法取构件的 ClassInfo
if (SUCCEEDED(hr)) {
      hr = (*pDllGetClassObject)(CLSID_ClassInfo,
                        riid, (void **)&pCIClassInfo_);
}
```

从以上的代码可以看出，获取构件 ClassInfo 的过程不需要其它的系统程序的参与，直接调用构件 DLL 模块的 DllGetClassObject 就可以了。

类信息所起的作用与微软 COM 的类型库相似，下面表 1 比较了类信息与微软 COM 类型库信息的不同。

	类信息（ClassInfo）	*类型库信息（Type Library）*
所属技术	“和欣”CAR 构件技术	微软 COM 构件技术
产生方式	CAR 文件编译结果	ODL 文件的编译结果
描述方式	构件自身完成	靠系统程序 OLE32.DLL 提取解释

表 1　和欣 CAR 类信息与微软 COM 类型库信息的比较

3.4　构件的导入信息（ImportInfo）

构件的导入信息（ImportInfo）指的是一种元数据。在 CAR 技术的构件封装中，把一个构件对其他构件的依赖关系作为构件的元数据即构件的导入信息封装在构件中。

为生成构件导入信息，CAR 构件编程技术要求在 CAR 构件描述语言中定义的构件必须指定它的 URN（Uniform Resource Name）。URN 是一个字符串，这个字符串类似于 URL（Uniform Resource Locator），是关于构件 DLL 文件的网络定位信息，在 CAR 中对 URN 的定义示例如下：

```
[
version(1.0),
uuid(e363b985-8a3a-40a6-b88c-b2e10274fe54),
urn(http://www.koretide.com/car/samples/hello.dll)
]
component Hello {
      ……
}
```

通过这种定义，URN 起到了唯一标识一个构件程序的作用，构件程序里的构件类由 CLSID 来标识。为了对构件文件进行快速定位，CAR 对 COM 标准的 CLSID 进行了扩展，引入了 CAR_CLSID，CAR_CLSID 除了包含构件类的 CLSID 外，还包括构件程序的 URN。其 C/C++定义如下：

```
typedef struct    CAR_CLSID {
      CLSID clsid;
      WCHAR *urn;
}     CAR_CLSID;
```

由于在 COM 的标准 API 及接口中，对 CLSID 的参数传递都以引用或指针方式传递，CAR 把 CLSID 放在 CAR_CLSID 前，就保证了对标准 COM 的兼容性。也就是说，CAR_CLSID 既可以在 CAR 平台上使用，也可以在微软 COM 平台上使用。除了 URN 外，对构件导入信息的描述还包括构件的版本号、ClassInfo 的版本号，最后修改日期，更新周期等，这些信息在构件升级及错误恢复时使用。

CAR 技术通过对构件的 C/C++源程序的预处理生成构件导入信息。当构件客户使用一个构件时，必须通过#import 预处理语句导入构件的定义。如：

```
// client of component - hello.dll
#include <stdio.h>
#import <hello.dll>
int __cdecl main()
{
      ……
}
```

在 CAR 编译环境里，调用 C/C++编译器编译 C/C++程序之前会先调用 CAR 工具 mkimport.exe 对 C/C++程序中的#import 语句进行预处理。

在经过对 C/C++源程序的预处理后，新生成了三种文件：

1)　临时的 C/C++源程序文件。与最初源程序的区别是把#import <xxx.dll>语句替换成了#include <xxx.h>。因为 C/C++编译器并不能正确处理#import <xxx.dll>语句，生成的临时源程序是提供给 C/C++编译器真正编译的。

2)　对源程序中的每一个#import <xxx.dll>语句，预处理器都生成了一个关于 xxx.dll 构件定义的头文件 xxx.h。头文件中声明了构件的 CLSID、URN、接口等等信息。

3)　当前程序的导入信息文件。导入信息文件中记录了当前程序所使用到的所有构件的导入信息记录（URN、版本号、更新周期等等）。

由此，构件的导入信息分成了两部分，第一部分是 URN，基于效率上的考虑，URN 以 CAR_CLSID 的形式直接声明在生成的构件头文件中，每个 CLSID 的后面都跟了一个 URN 字符串。比如，前面例举的构件类 CHello 的 CLSID 的声明如下：

```
const CAR_CLSID CLSID_CHello = \
{{0x3D19BC4C,0xB2C7,0x4EA5,
{0x84,0x09,0x63,0xDB,0x93,0x0A,0xD1,0xB7}},
L"http://www.koretide.com/car/samples/hello.dll"};
```

导入信息的第二部分是关于构件的版本、最后修改日期、ClassInfo 版本、更新周期等信息。由于这些信息并不是经常使用，它们被写入到一个导入信息文件，并且同一可执行程序的所有 C/C++源程序的导入信息在链接之前会被合并成一个。最后在进行链接时，程序的导入信息会作为资源数据链接到 DLL 或 EXE 文件的资源段中。

3.5 导入信息的使用

CAR 构件技术中使用了构件导入信息中的 URN，构件客户运行时使用的 CLSID 实际是 CAR_CLSID，如创建 CHello 构件对象的程序：

```
hr = CoCreateInstance((REFCLSID)CLSID_CHello, \
        NULL, CTX_SAME_DOMAIN, IID_IHello,
        &iHello);
if (FAILED(hr)) {
        ……
}
```

在 CoCreateInstance 中传入的 CLSID_CHello 是一个 CAR_CLSID，系统可以自动根据 CAR_CLSID 中的 URN 信息找到并装载正确的构件程序，并根据真正的 CLSID 找到相应的构件类。通过 URN 机制的引入，构件程序不需要安装、注册过程也同样能够被构件客户端使用，并且由于 URN 是在编译时链接到构件客户程序的二进制代码中，只要结合数字签名等安全机制，即使是管理员也无法修改构件的 URN，对病毒和黑客程序能起到有效的防范作用。

在"和欣"网络操作系统上，所有的构件程序都存放在系统的构件缓存目录中，目录中的构件程序以 URN 为唯一标识。当客户指定的构件程序不在系统中时，"和欣"操作系统将自动根据构件的 URN 到网络上下载构件程序到系统构件缓存目录中。正是通过 URN 等构件导入信息及构件缓存机制的引入，使得只要具备基本的构件运行环境，CAR 构件或构件客户程序就可以自滚动的运行。如：构件 A 依赖于构件 B，构件 B 依赖于 C。在某系统中最初只安装了构件 A，在构件 A 运行时，构件 A 在创建构件 B 的构件对象时，通过 CAR_CLSID 指定了构件 B 的 URN，系统就可以自动到网络上下载构件 B 的程序。同理，在没有事先安装构件 C 的情况下，构件 B 也能够得到正确运行。这种自滚动运行机制给了软件的使用者极大的方便，软件的使用者根本不需要了解除了他直接使用的软件之外的任何信息。软件的开发者也不再需要费心尽力的去为一个庞大而关系复杂的软件制作安装软件。此外，在"和欣"操作系统在启动后，一个构件第一次被装载时，系统会从构件的资源段取出构件的 ClassInfo，并根据 ClassInfo 中构件的更新周期判断构件是否需要升级，如果更新周期到了，系统会自动依据构件的 URN 到网络上去下载构件的更新版本。

4、 结束语

近年来，软件的可复用性受到的关注不断增加，正说明软件复用的问题没有得到很好解决。实现软件可重用性的关键就是软件构件中必须含有计算机可以理解的信息，就是自描述信息。计算机先通过解读这些信息来建立一个适当的运行时环境，然后在这个运行时环境中执行程序。如果构件的自描述信息能够被不同的计算机所理解，那么软件的可重用性问题就解决了大半。目前，在硬件驱动程序方面，软件的可重用性就没有得到充分体现。这直接导致移动设备无法做到无法达到第三代因特网的要求，网络新时代的到来于是受到牵制。而在带有充分自描述信息的构件化驱动程序的支持下，硬件设备的即插即用成为可能。游客就可以随手利用手机向任一自动售货机发送指令购买饮料，办公人员就可以把陌生环境中的一台打印机接到自己的笔记本电脑上随即开始打印资料，第三代因特网所提出的不论何地、不论何时、不论何人都可以享受信息时代服务的要求就可以达到。

CAR 构件技术实现了对 COM 技术中自描述特性的完善和深化。对于面向 WEB 服务的应用软件开发，对于实现硬件的即插即用，对于实现新体系结构的构件化操作系统，对于促进新一代网络应用的实现，CAR 构件技术均具有重大的应用价值。

参　考　文　献

1 陈榕.下一代 IT 技术焦点：信息交换[N].计算机世界 B12，2002.4

2 [美]Martin Gudgin 著，宋亚男译，IDL 精髓[M]，北京：中国电力出版社，2002

3 [美]Dale Rogerson 著.杨秀章，江英译.COM 技术内幕[M].北京：清华大学出版社，1999

4 Chen Rong. The application of middleware technology in embedded OS[C].In: 6th Workshop on Embedded System, In conjunction with the ICYCS, Hangzhou, P R China, 2001-10

5 [美]Nalini Venkatasubramanian.Safe 'COMPOSABILITY' of Middleware Service[M].IEEE, the professor of University of California-Irvine, 2002