计算机科学 1996 Vol. 23 №. 1

面向对象技术中对象模型的复杂性

麦中凡　刘书舟
（北京航空航天大学　北京 100083）

摘　要　In this paper, the authors analyse the diversing of the existing object model. Now, about 300 kinds of can be made up of with semantic components. The authors points out that it is this diversity that is the intrinsic reason why there is no clear view about OO technology. At the end of the paper, the authors prospect the unification of various existing object models.

关键词　OO model, Interoperatability, Data exchange, Open software environment Heternetwork

1. 引　言

80年代多媒体应用，网络技术史就，以及各大公司推出的大量计算机辅助软件工程工具(CASE)，把OO技术炒得火爆。似有90年代OO技术将会像80年代结构化程序设计那样占据软件行业主导地位。但情况并非如此，OO技术在自己前进道路上也遇到不少麻烦，以致于至今尚无规范化的OO软件开发方法学，OODB无法和RDB匹敌，甚至连一个公认的对象模型都未统一。各行各专业都吸取了OO思想改善本专业的软件，五花八门的对象模型使OO技术支持重用、移植、可维护性、可互操作性等原有初衷大为削弱。所以88年以来，美国OO厂商和标准化组织都积极行动起来成立OMG, ODMG等专业组，以求统一协调OO技术标准，但难度相当大，发布的标准往往只能规范比较核心的部分。本文试图说明对象模型内在的复杂性，并分析了当前对象模型标准化的动向，以及对象模型统一的前景。

2. 对象模型的变异

"纯"面向对象模型来自Smalltalk，除了充分体现封装、抽象、继承、多态(动态束定)等优越性而外，与生俱来的无类型和动态消息匹配使程序运行效率低下(较C语言模拟实现慢30倍)。单继承限制了表达能力。所以各专业在引入面向对象技术时，并未以Smalltalk作为首选工具。当然最初以传统语言模拟Smalltalk以求高效，典型的例子是Objective C。但更多的软件工作者，以本专业沿用已久的传统语言扩充OO功能，如Object Pascal, Object Cobol, Object FORTH, O-OP, PCLOS, C++。其中最有成效的是C++。由于采用超集和原有语言兼容，以利于原有软件不致重写的策略，不得不对Smalltalk的最初对象模型有所改变。例如，过程语言的扩充者都以过程和函数调用代替消息发送，无类型的属性增加了类型，从而造成类、类型不分的语义，以保留高效类型检查的强类型优势。静态编译的强类型带来了一系列新语言特征：虚函数、友员、重载运算符、以及精确的可见性控制。此外多数语言扩充了多继承，以便更方便地模拟客观世界。这都是造成对象模型变异的语言方面的因素。

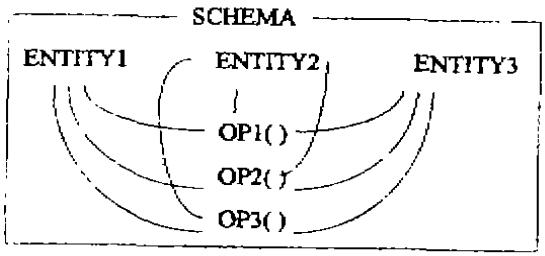

图1　CAD中典型的对象

此外，专业不同，用途不同对对象模型的要求也不同。如CAD领域，它的复杂对象是直接描述的简单对象聚集，称为Schema，相当于无类型功能的程序包，简单对象entity却是有实例的类，但不包含操作，一个Schema中有封装的操作，它与简单类型entity并列，并为Schema中的entity共享，示意如图1。Schema之间可以引用(即复杂对象间的"继承")，它在Schema定义中显式指明，凡先此定义的Schema(可放在库中)均可线性联接，被联接Schema的entity和op可直接为本Schema的op所用。无预

定义继承体系。而 entity 却可以是一继承体系中派生的子实体(类)。

CAx 最关心的是对象在整个加工过程(从模型客观对象——CAD 图——CAPP 作出工艺——CAM 作出数控机床的控制卡)中(我们说是沿时间垂直进程)语义描述的一致性,和各加工阶段间数据交换的完整性。

以纯 OO 观点解释 CAx 中对象模型:Schema 是封装了子对象和操作的实例(本身不是类和类型)。entity 相当于类可以产生实例,但又不封装操作,entity 有 OO 意义的继承,但只继承数据。Schema 间的"继承"靠人为指定,指定后即可引用操作。

分布式网络环境中,关心的是对象的协作和通信,即空间域上水平(各结点间)的可互操作性。希望封装对象直接传输,因此过于复杂的继承关系不利于派送(分布)和传输。为了简单,每个网络结点都封装为大对象,不是客户就是服务器,等同(Peer)对象既是客户,也是服务器就比较复杂了,Peer 到 Peer 交互是正在攻克的难点。

开放式软件工程环境中,希望不同厂商制造的各种软件工具可以任意集成,其集成模型是抽象为单个对象(很少有 OO 意义的类对象,因为即使一个环境上需要多种语言的编译,目前还做不到写一个抽象的编译器,实例化成各语言编译器)作为软插件插入,它既关心时间上垂直的可互操作性,即不同开发阶段工具(如编辑、编译、连接、编码、测试工具)的数据交换,也关心地域上水平分布的可互操作性(异质网上的分布式环境)。更重要的是包英式封装,即各厂商以不同语言在不同结点的机器上运行的工具封装为集成界面上可统一操作的对象。因此不得不通过对象要求代理器(Object Request Broker 简称 ORB)接口把自己对象化,因此也不希望复杂的继承关系。

作为软件工程环境的支持,数据库系统则希望有完整的多继承体系(完善的数据模式)以支持信息工程(J. Martin 提出的针对企业管理模型下的利用大量重用件)的开发,不仅有纵向的类继承体系,还要有横向查找索引的关系描述。为了保证对象数据库的完整性、共享性、一致性,还需要事务、约束、触发,版本等机制,数据库中对象因而更加复杂了。

多媒体应用,特别是非正文的,一般数据量大且数据全靠与其对应的操作才有语义(如扫描图象的二进制位模式(点阵)大块 BLOB)。为了等价媒体切换,往往作成引用子对象,即一个对象是某格式正文的,且包含指向语义等价(字体、格式不同的正文,或图象、声音等)的子对象指针。多媒体的对象除 CAD 图形外,一般继承关系不复杂。但 Hyper Text,HyperMedia 的基本思想是联想,即动态索引检索,以便在海量信息中快速查找与切换。目前联想交给用户,由用户指点图标图符切换,因此完善的联想即智能检索在方法中增加了推理用的规则、约束、触发。

人工智能应用中知识表示的框架模型和多继承的对象模型基本一致,但框架中的槽比较自由,可以是数据,也可以是约束、规则和操作。当然框架中的谓词一般对应为方法。

总之,对象应用的广泛性带来了对象模型的变异。

3. 对象模型再分析

如果暂时不考虑完全动态委派(delegation)的只有对象实例没有类对象的那一分枝,我们只说类-实例模型中的类对象,分别从属性、方法、继承、封装四个方面分析已出现的差异。

3.1 属性

对象的属性原本只有基元(纯量)类型和结构类型(表、数组、记录)数据,以其值表征该对象的状态,我们称它为(本)对象属性。当然,还有继承自父类对象的对象属性,则称继承属性。从语义上它们虽无区别,但操纵和表示的语法上确有区别,还因为有重载(例如,a,p1::a 是可覆盖的两个属性)。除此之外,对于复杂或多媒体对象还有三种属性:

(1)包容子对象(属性)。严格说来,聚集的对象属性被包容于某对象也就是子对象,不过其中的操作被隐含。例如,整数类型隐含的整数操作一般不显式写出,表现为只有变量(数据)。我们把显式写出操作的属性对象称为包容子对象,是对象的聚集成员,它在对象实例中生成自己的子实例,并为对象包容的方法成分使用,如图 2。包容子对象一般不指明对本对象继承体系以外的继承,否则语义关系太复杂难以控制。

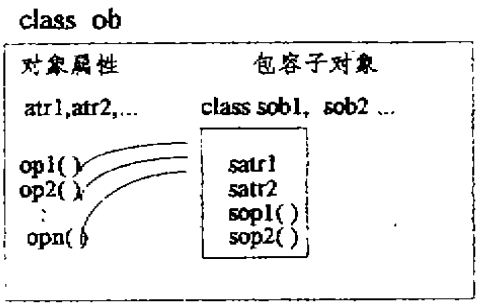

图 2 包容子对象

(2)引用子对象(属性)。包容子对象是对象内部不独立的子对象,其实例只为包容对象内部使用,其它对象无法令其生成独立的实例。引用子对象则包

容了独立的子对象，例如，我们定义了一个引擎类和一个车轮类，Class Engin，Class Wheel，在定义小汽车类时可利用它的实例：

```
class Car{
        Engin V115;//引用子对象
        Wheel DC[4];
        //…
};
```

而 Engin 和 Wheel 均有自己继承体系中的位置。当然，对于 C++之类语言，引用子对象可用指向类的指针实现。

(3)二进制属性。多媒体中二进制大对象块 BLOB 是特定语义下的属性，没有程序语言中的类型(无论是动态还是静态的)语义，不作类型检查，但和数据方法联系紧密，并有特殊的输出。

3.2 方法

对象的方法从内部角度而言是对属性的操作，就对象的外部特性而言它表征了对象的行为，也是外界与对象通信的接口，是封装对象的窗口。目前对象模型中有以下方法：

(1)行为方法　凡与外界通信的方法都归之为行为方法，因为它们要改变对象的状态(属性的值)。行为方法包括继承来的行为方法。同样，它的语言实现要有分辨重载和动态匹配的机制，但语义一样，我们归于一种，也可以列为两类方法。

(2)数据方法　凡导出属性值的方法均列为此类，例如，圆心的平面座标，加上半径三个属性值不一定是圆，只有加上 circle.draw( )(画圆)的操作，才能描述圆的一切性态(可导出圆周上所有座标、周长、面积)。再如，有整数 0,1 两个属性值，加上 succ( )(后续)函数，则可导出所有正整数，调用 circle.draw( )和 succ( )实则引用数据。CAx 中这种表示法是相当多的。

(3)描述对象性质的方法　调用这类方法一般不改变对象状态，因而也不表明对象的行为，充其量是行为的约束。这类方法有两种：

·规则/约束。语法上和方法无异，语义上是检查(但不改变)属性值，返回一真假值。多用于对象内部作其它方法的卫式条件，如果卫式条件(不)满足则触发另一方法，或引发异常。

·关系，严格说对象间的继承也是关系，描述了对象静态继承体系的纵向联系，但对象数据库中为了提高效率和给出更多显式语义信息，扩充了关系，即由程序员指明对象体系中对象间的横向联系。例如从人的父类可导出应届高中生、个体户、军人…可以在学校类中指出他们具有 classmate(同学)关系。假定一个学生只在一个学校就读，下面反映了学校与学生之间一对多的关系：

```
class School{
  set(Person)classmate inverse Person::sch;
  //…
};
class Person{
  School sch inverse School::classmate;
  //…
};
```

(4)共享方法　由于封装，一个类的方法仅能加工本类的数据(即属性值)。然而，有许多应用两个或多个类要用到同一个方法。例如矩阵和向量混合运算是十分常见的，我们虽然可以在它们的共同父类中定义算术运算的各个方法，并施以技巧完成此种混合运算，但破坏了矩阵类、向量类语义直观性，也不利于维护。C++采用 friend(友员)函数机制，打破封装，实现方法共享。

3.3 继承

单继承是每一对象严格只能有一父对象，反过来可以有零到多个子对象。这种继承体系的拓扑图是一颗树。它的优点是结构清晰，派生、查找容易，但表达能力弱。

多继承是一个对象可以继承多个父对象的属性和方法。显然，继承体系的拓扑图是一张网，表达能力强，但方便、灵活性也带来了不安全性，因为有可能继承自己的子孙或陷于继承本身的死循环。于是规定了有向无环的网状拓扑图，解决了直接间接自继承问题，但分辨继承路径的算法开销仍然不小。因此，两种继承方法并存。

3.4 可见性控制

封装源自把与外部无关的内部数据隐藏起来，以免引起错误指令误操作，其客观效果是提供了易于模拟客观世界对象的程序对象实体，增加了显式语义。我们把封装界面上对外开放的属性和方法称为公有成员，则其他自然是对象私有的。这是承袭模块式语言 Ada，Modula-2 的机制。Smalltalk-30 没有私有成员概念，所以不宜编制大系统。可见性控制减少了复杂性，它的原则是每个对象可自由控制外部可见性且越小越有利。当有继承时派生对象似乎不太自由。一般说来，父对象中私有成员在派生类中仍私有，公有仍公有。若继承 25 层(随便举例)那么公有部分加起来怎么也不会少；能不能人为减少？这是第一个问题，再者个别父对象的私有成员想变成派生对象的公有成员有没有可能？这是第二个问题。于是，C++提供了私有继承和受保护成员概念以保证程序员可见性控制的充分自由度。

受保护成员在父对象中等于私有成员，在派生对象中等于公有（外部可见）成员，还要加上公有继承的先决条件，这就解决了上述第二个问题。

在私有继承的模式下，父对象中所有成员无论公有、私有、受保护的一律成为派生对象的私有成员。这就为解决第一个问题提供了保证，只将希望保留外部可见的重写于派生类。做法是，定义一个外部可见成员，其函数体为调用私有继承来的那个公有成员。

三种成员，两种继承模式，则对象成员有六种可见性的访问权限。

4. 统一的对象模型

我们按语法归类分析了已经出现的类对象的语义成分，它们可以组成：5 种属性×5 种方法×2 种继承×6 种可见性＝300 种语义成分的对象模型，还不包括具体语言在实现对象模型时的语法细微差异（如同一机制用了不同关键字），也没包括 CAx 中基于对象的 Schema 模型，可见统一面向对象的对象模型的复杂性和难度。

通过以上分析，面向对象并不像结构化程序设计那样简单：只要坚持嵌套的严格一个进口一个出口的程序模块就可以了。大家都作面向对象程序，大家心中对象模型都是不一样的。然而时代又不允许分封割据，异质网上的开放软件环境又要把大家集成起来。这个矛盾极端深刻，本来一个好好的面向对象范型就是代替不了传统的"落后"的东西。更深刻的是互不兼容的软件产品反过来损害新技术开发者的商业利益。于是，大家只好暂停，先统一对象模型标准。88 年后国际上各专业陆续推出对象模型标准，但距统一模型还相当遥远。首先 CAx 领域是一个先于 OO 技术之前就在寻求标准化的领域，并有其历史体系，OO 思想和抽象数据类型使该领域得以实现垂直方向的数据交换。早期的数据交换标准 IGES(1981)主要针对图形，缺乏产品全生命期的数据，其规格说明又有二义性，使数据交换不畅，因此，ISO 发布了 PDES/STEP 产品数据交换标准(1988)。为满足 CAx 的要求，专门制定了 EXPRESS 语言，其对象模型是上文所说的 Schema 模型。为操纵模型的实例，又提出 SDAI 标准，将 EXPRESS 的模型转换为具体对象语言模型，即按 SDAI 联编到具体语言。

美国的软件工程研究者和 CASE 厂商 88 年成立的 OMG 组提出了 OMG 的对象模型以及包装非对象软件工具和数据的 ORB 标准。91 年后开放式软件环境向异质分布式网上扩展，则要求有一公共的各结点均接受的 ORB，即 CORBA(Common Object Request Broker Architecture)标准，CORBA 是 OMG 对象模型在分布式网上的接口标准，而应用领域的目标可以从 EXPRESS 到 SDAI 到 CORBA。这样，虽不能统一模型，但通过标准接口的转换，也可以实施垂直和水平数据交换。

91 年美国数据库业界成立 ODMG 组，在 OMG 核心模型基础上扩充了数据库需要的持久性、查询、事务、关系等机制，发布了 ODMG-93 标准。ANSI 和 ISO 也许吸收了 PDES/STEP 标准的经验，在技术快速变化的情况下持观望态度。所以，ODMG-93 只是行业标准。OMG 组在设计模型时，采取了一种可扩展方式，即只定义最小核心，各专业在它基础上扩充，如图 3 所示。模型和实现采用联编(Binding)接口，这样可不受具体语言机制发展的影响。

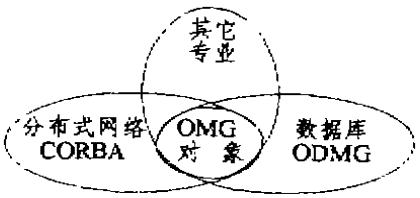

图 3 OMG 模型和其它标准的对象模型

由此看来，统一的对象模型还有待时日，它将是所有专业标准的超集，即图 3 的包络。现在是由核心向各专业扩展，一旦成熟，各专业标准的模型都将是无语义冲突的子集。本文对各种语义成分的分析不过是为研究语义成分冲突提供素材，也为广大业者理解当今对象模型发展提供思路和线索。

参考文献

[1]R. M. Soley, An Object Model for Integration, Computer Standard & Interface 15(1993)

[2]计算机集成制造(CIMS)约定、标准与实现指南，兵器工业出版社，1994. 9

[3]张学平，面向对象的程序设计的演化特征及应用，计算机研究与发展，1994. 4

[4]Thomas Atwood, ODMG-93 Standard, Object Magzine, Sept.-Oct. 1993