支持对象间关系的程序设计语言研究*

万建成 张曙明

(山东工业大学计算机科学技术系 济南 250061)

E-mail: wanjch@dms.sdut.edu.cn

摘要 在论述了面向对象技术中对象间的关系作为第一级建模概念的重要性之后,该文设计并实现了显式支持对象间关系的RCPP(relational C++)语言.它提供了显式描述对象间关系特性和语义的机制,利用关系来动态地控制对象行为的作用和传播.它的运行是通过一个转换器把RCPP代码翻译成C++代码,再经C++环境编译后,形成可执行程序而实现的.文章对RCPP语言的模型、语言提供的服务以及具体系统的描述和实现作了深入阐述.

关键词 面向对象,对象间关系,程序设计语言,面向对象建模.

中图法分类号 TP312

1 研究意义及RCPP系统

面向对象方法的出发点是建立可靠、可重用的对象.然而,对象并不是孤立存在的,对象之间必然存在着这样或那样的关系.只有通过保持和维护与其他对象间特定的关系,对象才能实现特定的功能.对象间的关系正在软件设计中发挥着重要作用.在分析阶段,关系是首要的、必不可少的建模元素[1-3];在设计阶段,关系的显式描述可以使系统的结构清晰、简洁[4];在实现阶段,关系的显式实现可以方便并简化繁杂的程序工作;在后期维护阶段,显式化的关系使程序易于理解、修改和完成故障检测.因此,在面向对象软件开发的全过程中,都应该把关系作为与对象同等重要的概念加以使用.

本文从对象间关系的设计与实现出发,研究设计了程序语言RCPP(relational C++).这项研究的动机源于智能CAD等复杂软件的开发和面向对象分析设计的研究.在复杂系统中,对象间存在着复杂的结构和行为上的关联.在现行的面向对象程序语言中,只能以隐式的方法描述和实现这些关系.这就降低了系统的结构清晰度和可维护性.

RCPP以C++语言为基础,通过引入新的语言成分来方便地支持对象间关系的建模,实现了对象间关系的显式表达、特性描述和动态维护.与面向对象中已有的继承、成员和友员等关系不同,RCPP中的关系是指对象的状态和行为之间存在的关联性.它为表达对象间以及类间的行为关系提供了有效、清晰、方便的描述手段.RCPP中关系研究的独特之处在于:归纳了对象间关系的抽象描述;抓住了相互作用的对象间的、除单纯可达性访问外的数据约束和行为依赖的特性;提出并实现了对象关系的形式化语言描述.

2 对象间关系的研究现状

在面向对象技术中,对象间关系作为一项重要的概念,无论在分析建模方面,还是在语言实现方面,都还处于研究阶段,并正在得到广泛的重视.

2.1 对象间关系的分析建模

这方面的研究主要集中在面向对象的分析设计OOAD(object-oriented analysis and design)和面向对象数

* 本文研究得到山东省自然科学基金(No.93G8379)资助.作者万建成,1949年生,教授,主要研究领域为人工智能,面向对象技术,自然语言处理.张曙明,1973年生,工程师,主要研究领域为计算机软件,面向对象技术.

本文通讯联系人:万建成,济南 250061,山东工业大学计算机科学技术系

本文1999-01-19收到原稿,1999-05-12收到修改稿

据库 OODB(object-oriented database)[5]中,OOAD 的目标是对应用领域问题建立对象模型,标识和建模对象间的关系以及它们之间的相互作用.客观世界中的实体是多种多样的,实体之间的关系也是复杂的;同时,关系所涉及的对象是在不断变化的,甚至关系本身的特性也可能发生变化.面向对象分析与设计的任务就是对现实世界进行描述,对实体进行抽象,构造出对象及它们之间关系的模型.例如,在 CAD,CASE 等领域中,不仅需要对象属性和方法的封装存储,还需要管理具有复杂关系的对象,要求对象之间的关系保持丰富的语义.在这些应用中,对象间引用复杂、相互依赖,需要以关系来描述对象的复杂语义.因此,在面向对象的方法中,关系对于组织对象起着非常重要的作用.目前的 OOAD 方法[4,6]主要有 Shlaer&Mellor 方法、Coad&Yourdon 的 OOAD 方法、Rumbaugh 的 OMT(object modeling technique)方法等.在这些方法中,关系都是必要的建模概念.OOAD 对于对象间关系的研究主要是在关系和属性的分离、关系语义的捕捉、关系的特性及其表达、关系的类型及其标识等方面.

总之,目前 OOAD 方法都把关系作为基本概念,它对系统构造和描述起着重要的作用.但是,在这些研究中基本沿用了数据库的 E-R 模型对关系的认识和处理,它主要是基于关联对象可达性访问的,缺乏应用问题逻辑蕴涵的对象依赖关系的描述和自动维护.

2.2 对象间关系的语言描述

在 OOAD 建立起系统模型以后,需要通过面向对象语言 OOPL(object oriented programming language)加以实现.从现行 OOPL 上看,对象被其名字、属性和方法以及产生的过程清晰地确立了它在系统中的存在,而对象之间的关系则隐含在属性、方法等代码中.这是因为传统的 OOPL 缺乏对象间关系的描述机制.用户除了在方法设计中控制对象之间的关系外,没有其他手段可用来直接表达和控制对象之间的关系[7~9].

当前,支持关系的 OOPL 研究方法主要有两种.一种是不改动语言,而引用专用类库来支持关系.如 ILOG 公司提供的 ILOG Server 类库[10],它允许实现 OOAD 中常用的关系概念,支持描述复杂对象结构中对象间关系.另一种是引入少量新语言机制,扩充类模型.如在青鸟 II 的 CASE C++[11]中,增加了永久对象间的 LINK 关系声明来实现对象间的关系.

ILOG 类库中的基础类有 10 多个,类与类之间的联系复杂,加之并不是一种显式的关系描述,给用户的理解和运用带来了困难.CASE C++的对象间关系只是针对永久对象,而且没有涉及关系中的状态和行为语义.总之,目前的研究成果中还存在着以下不足:关系的复杂特性、复杂语义描述不够直接或无法描述,对于关系的动态性也没有有效的表达手段.

3 RCPP 中的对象间关系

3.1 RCPP 系统概述

RCPP 是在 C++之上开发的,它有两个组成部分:RCPP 的语言机制和翻译系统.

RCPP 中用关系(relation)来规范对象间关系及其行为.关系是为显式规范对象之间的行为关联而制定的一种结构,用来形式化对象之间的合作与行为联系.一个关系定义了一组通信的参与者及其语义.关系语义扩大了通常的类型特征和 OOAD 中关系的可达性概念.它包括数据和行为的约束与关联以及多个对象间属性和行为的依赖.

RCPP 把关系从对象的属性和方法的定义中分离出来,通过显式的语言表达机制进行定义.用户只需简单地定义关系,而关系的维护完全由系统来完成.RCPP 是关系与面向对象技术的结合.与类和对象一样,关系是第一级的语言概念,它是类之间状态和行为联系的抽象.关系独立于面向对象的原有概念,但是它与对象、属性和方法有机地结合在一起.另外,RCPP 语言遵循 C++的语法规范,在风格上与其保持一致.

3.2 关系的描述要素

关系的描述要素包括:关系的类型、基数、方向、引用标志、可视性和关系变量等.

关系类型是理解对象的组织构成和相互作用模型的基础,它确定关系参与者目标方的类型并确定标识关系的管理方法.

关系变量通过关系类型的实例来建立.这就约束了满足参与者关系语义的具体对象.通过C++原有的语言来构造关系的实例,就可以显式地建立关系变量.

关系的基数是指在两个相互关联的对象类中,一个类中的对象实例和另一个类中的对象实例相关的数目.按照基数特性,对象间的关系分为一对一、一对多、多对一和多对多.RCPP直接支持一对一和一对多这两种关系.

关系的角色是指关系的端点.角色提供了一种观察关系的方式.二元关系有两个角色.每个角色都有一个名字,它唯一地标识关系的一方,可以是一个对象实例、一组对象实例的集合和一个对象类.名字的确定非常重要,它要清楚地表达对象在关系中发挥的作用.角色名对于区分两个相同对象类之间的两种以上的关系更为重要.

关系的方向性是指关系是在一个方向还是在两个方向上发挥作用,它指关系语义的作用方向.由此,关系分为单向关系和双向关系.

关系的引用标志是指关系的独立/依赖以及共享/排他标志.这两个标志与对象的生命周期范围、对象本身和对象之间关系的一致性有关.独立/依赖关系是指一个对象的存在是否依赖于另一个对象.共享/独占是指一个对象类的对象实例是否可以同时与另一对象类的不同对象实例发生相同的关联.

3.3 关系的语义

RCPP中的关系声明不仅提供了关系特性,还提供了关系语义直接显式地描述.关系语义的显式描述是RCPP的一个重要特点,它提供了复杂语义关系的描述机制,从而大大提高了系统建模的表达能力.

关系语义是指关系的参与者所要维持的数据约束和行为依赖,它是关系具体含义的表达.RCPP中的关系语义扩展了OOAD中关系的可达性概念,分为4种:属性相关、行为连接、附加属性和附加方法.其中属性相关关系又分为逻辑约束和赋值约束.

用户在关系语义声明时可以加入关系自身特定的属性和方法来体现关系所表达的特定意义.这种语义称为附加属性语义和附加方法语义.例如,在两个地点对象"北京"和"上海"之间的"航班"关系中,加入"距离"、"航行时间"、"票价"等属性和"飞行"方法等.

属性相关语义是指关系的目标对象的一个属性值和源对象的一个或多个属性值之间存在某种函数关系.这种函数关系包括逻辑关系、赋值关系和导出关系.

逻辑关系是指两个数据之间存在着"<,<=,==,>=,>,!="中任意一种关系.逻辑关系是双向作用的,具有逻辑关系的两个数据既是激发值,又是被激发值,一个数据的变化都会引起另一个数据的变化,用以维持逻辑关系所表达的含义.例如,形如"$a<b$"的逻辑关系,当a被重新赋值时,需要检查赋给a的值是否小于(<)b的值.如果是真,则赋值有效;如果为假,则要求的逻辑关系未能满足,赋值操作被取消.同样地,对b的赋值也要作同样的检查.

赋值关系是指形如"$a=f(b)$"的关系,它说明a和b之间存在的函数关系.赋值关系是单向作用的,其中a是被激发值,b是激发值.b值的变化必然会引起a值的变化.

在属性相关语义说明中,对于那些完全依赖于其他属性值的属性,称为活动值或导出数据成员[10].导出数据成员所依赖的属性称为项目值或项目数据成员.项目数据成员的变化必然引起导出数据成员的变化,它具有激发功能;导出数据成员不能直接被修改,只能随着相关项目数据成员的变化而变化.这种关系称为导出关系.如在"工作"关系中,"职员"的"奖金"是导出数据成员,它不能被直接赋值,只能由项目数据成员"公司"的"利润"值引起变化.导出关系是一种特殊的赋值关系.

行为连接语义是指,关系的源对象的某个方法调用必然引起目标对象的某一方法调用,像这样来形成方法调用链.如在"工作"关系中,"公司"对象的"需求"方法调用可以引起"职员"对象的"答复"方法调用.

那些只能由其他方法的调用而被激活的方法,被称为导出方法.导出方法所依赖的方法称为项目方法.项目方法的调用必然引起导出方法的调用.导出方法平时处于睡眠状态,不能被其他方法直接调用,只有当相关项目方法被调用后才被激活、运行.

由于以上显式语义表达的引入,RCPP中的关系不仅可以表达对象之间简单的引用关系,还可以方便地表达对象之间在状态和行为上的复杂语义联系.

3.4 关系的描述和操作

关系的描述与操作分为以下几个步骤：

(1) 关系类型的声明.关系类型的声明是关系描述的第1步,它是可重用的.关系类型用以确定关系的基本要素,包括关系的目标对象类型、关系的基数、关系的方向以及关系的引用标志等.关系在类型声明时并不确定关系的语义.

(2) 关系变量的声明.关系变量是关系类型的实例,它唯一地确定了一个关系.关系变量是引用该关系的标识.只有在对象类内部声明时它才会有效.

(3) 关系语义的说明.关系语义是对关系的进一步解释,它具体确定关系的参与者之间存在的数据约束和行为依赖.关系语义和关系类型是相互分离的.

(4) 关系的操作.可以根据系统提供的关系标准接口对关系的要素进行操作.这包括关系目标对象的添加、删除、引用、关系基数、方向及标志位的访问等.

3.5 关系的继承

在RCPP中,关系作为类的成员,与类的其他成员一样,也是可以被继承的.也就是说,父类中声明的关系将被自动地继承给子类;如果不加额外声明,子类就可以直接使用父类定义的对象关系.当然,子类中仍然可以声明新的关系.另外,关系继承的可见性可以通过C++的public,private,protected这些关键字来控制.对于后两者,由于关系变量对外不再有效,因此需要设立特别的用户方法来转换、传递对关系的访问和操作.

对象关系的表达和实现可以有两种方法[12],它们的描述运行机制差别较大.一种是外挂式,即先声明特别的关系类,然后将关系类作用到发生关联的对象上,通过关系类对象管理发生关联的对象.另一种是内嵌式,即把关系声明为关系源对象类的成员.内嵌式直接利用了类的描述、继承、运行机制.目前,绝大多数关系研究采用的是这种方法.RCPP采用的也是内嵌式方法,虽然也有关系类型声明,但这只是一种实现上的需要.关系类型自身也具有继承性,继承的目的是形成附加属性和方法.

4 RCPP的描述与运行机制

RCPP是沿用了C++的面向对象模型的框架、在其类的基础上引入关系声明而发展起来的.关系声明使得RCPP在通常类型系统之上,为对象之间的属性约束和行为连接提供了直接、方便和灵活的支持.关系的应用状态的维护由C++的原有对象机制和新引入的关系管理机制共同来完成.

RCPP的对象模型形如:class类名{属性声明;方法声明;关系声明;}.

可见,关系与传统的OOP对象模型融合为一体,形成了RCPP的对象模型.它除了通常的属性和方法两大部分外,还包括第3个部分——关系.关系作为对象的一种成分在类声明中定义后,由RCPP通过支持机制维护其基本特性或语义,并可实现动态的操作.关系以类接口形式给出,对象实例可以动态地利用这些接口,完成所需的操作.

在RCPP中,关系是静态定义、动态维护的.即关系的特性是在类声明时静态说明的,而参与关系的对象则是在对象运行时动态确定的.关系声明是在关系的源类中加以定义的,也就是说,在类定义时就必须说明该类作为源类的关系接口.类的对象实例可以利用这个接口,也可以不用这个接口.当对象实例不用关系接口时,关系的目标对象链即为空,关系接口不发挥任何作用.只有当关系的目标对象链不为空时,关系接口才起作用.

为了实现关系的描述和运行,RCPP提供了以下几种新的语言机制：

(1) 新关键字和运算符.关键字relation用于在类声明中表示关系声明的开始.关键字RelatingTo用于声明关系类型.运算符　->>用于关系行为连接语义的说明.

(2) 支持关系运行的基础类.RCPP的实现机制是通过基础类AbstractObject和RelatingTo以及相关的转换方法来完成.AbstractObject是所有基本对象的抽象基类,系统最终将需要关系处理的类定义都由此抽象基类导出.该抽象基类设置是为了维护对象的生存性和引用完整性,它是采用类似OLE(object linking and embedding)中的相关技术来实现的.RelatingTo是一个基于模板的通用关系基类,是为了维护关系的基本特性和操作

而设置的.

(3) 关系声明语法.它包括了关系类声明、关系变量声明和关系语义声明.关系变量可以通过 public,private,protected 关键字控制关系的可见性.

(4) 关系操作语法.RCPP 提供的标准关系操作方法用于访问、操作关系及其特性.关系操作的基本形式是〈源对象〉.〈关系变量〉.〈关系方法〉(〈参数〉).关系关联对象访问的基本形式是〈源对象〉.〈关系变量〉[参数].

5 RCPP 的运行系统

运行系统是指把 RCPP 语言程序翻译、链接成可执行程序的系统.由于 RCPP 是对 C++语法的扩充,所以只需设计一个转换器,用它把 RCPP 程序转换成等价的 C++代码,然后利用 C++的编译、链接机制形成可执行程序.在整个运行系统中,RCPP 转换器是核心.它包括以下几个部分:(1) 读入 RCPP 语言程序;(2) 进行词法、语法检查,给出检查信息;(3) 分析程序定义(包括分析 RCPP 语法,确定关系意义、对象之间的作用);(4) 修改原有的程序定义(追加新属性,修改方法定义,转换接口,增加基类,根据上一步的分析结果把 RCPP 语法的关系声明转换成 C++程序);(5) 输出与 RCPP 程序语义上完全等价的 C++程序.

6 一个简单 RCPP 程序例

该例说明了 RCPP 中对象关系和关系语义的描述方法.

```
class Person;     class Company;
relatingTo(Person,"0..1",double) Couple;
    //声明与"Person"类对象发生关联、基数为"0..1"、方向为"double"的关系类 Couple relatingTo(Company,"*",double) WorkFor;
    //声明与"Company"类对象发生关联、基数为"0..*"、方向为"double"的关系类 WorkFo
class Person {
    private: char Name[20];      int Age; char Sex;      int Money;     int Salary;
    public: void GiveYouHelp( );      ......          //方法声明,从略
    relation:                                          //开始关系声明
        { Couple r1;                                          //关系变量声明
          {Sex!=r1[1].Sex;  Money--r1[1].Money;}              //关系语义声明
          class work for: public WorkFor{     //含附加属性和方法的关系定义
          public:
              int getSalaryFromProfit() {int x=0;     //附加方法
                  for (int i=1;i<=getUpBound();i++)x+=getTObject()  >Profit*0.0001;
                  return x;}
          private:   CString Position;                        //附加属性
          };
          work for r2;                                        //关系变量声明
          {Salary=r2.getSalaryFromProfit();
              r2[].AskForHelp(this)-->>GiveYouHelp();          }
      };    };                                                 //结束关系和类声明
class Company {
    private: int Profit;     ...;
    public:  int AskForHelp(Person *ps);     ...;
    relation: ...;  };
```

在该例中,Couple 和 WorkFor 是两个双向关系类.它们的目标对象分别是类 Person 和 Company,基数是 0..1和 *.类 Person 具有 Couple 和 work_for 关系,r1和 r2是它们的关系变量.在 Couple 关系中有两个属性相关语义.一个说明目标对象的 Sex 属性必须与源对象的 Sex 具有不同的值,它是建立合法的 Couple 关系的前提.另一个说明目标对象的 Money 属性必须与源对象的 Money 具有相等的值.当源对象的 Money 值发生变化时,目标对象的 Money 值也发生相应的变化,反之亦然.这是因为该关系是双向的.

work_for 关系是从关系类型 WorkFor 继承而来的,继承中附加了语义属性 Position 和方法 getSalaryFromProfit().该方法计算类 Person 的 Salary 值.Salary 是一个导出数据,它的值只有受在 Profit 值发生变化时的激发下才发生修改.在 work_for 关系中,当源对象向目标对象发出 AskForHelp()消息后,目标对象就会激发源对象的 GiveYouHelp()方法.这样,源对象的操作就与目标对象建立起了行为的依赖关系,而且这种关系具有自动和动态的传播特性.

7 结束语

通过扩充 C++的语法和语义,在原有对象模型上增添关系,RCPP 设计并实现了对象间关系的直接、显式描述和关系语义的自动维持.它通过引入新的语言机制和类支持,使对象间关系作为与类和对象等同的概念而得以实现.这提高了语言的描述能力,减少了对象间关系建模的复杂性,提高了涉及对象关系、尤其是动态对象关系的软件可维护性和可扩充性.

本文提出和设计的 RCPP 语言描述和运行机制,包括关系的继承、关系语义的自动维持等,都已经得到全面实现,并在有关小型软件设计中得到应用.进一步的工作包括:研究并比较关系的外挂式实现,研究多元对象间关系的组成和实现.这些将最终促进全面实现显式对象关系的描述和建模方法,为复杂对象系统的描述和开发提供简明、高效的软件技术.

参考文献

1 Rumbaugh J. OMT: the object model. Journal of Object-Oriented Programming, 1995,7(8):21~27

2 Rumbaugh J. Generating code for associations. Journal of Object-Oriented Programing, 1996,8(9):13~17

3 Rumbaugh J. Attributes and associations. Journal of Object-Oriented Programming, 1996,9(3):6~8

4 Gabriel E. Improving object-oriented analysis. Information and Software Technology, 1994,36(2):67~86

5 Li Long, Mai Zhong-fan, Cao Guang-tong. The implementation of relations among objects in OODB. Mini Micro Computer Systems, 1996,17(8):8~15
(李龙,麦中凡,曹广通.OODB 中对象关系的实现.小型微型计算机系统,1996,17(8):8~15)

6 Chen Han-sheng, Xu Zhi-chen. Object-Oriented Development Technique and its Application. Shanghai: Shanghai Science and Technology Documentation Press, 1995
(陈涵生,徐智晨.面向对象开发技术及其应用.上海:上海科技文献出版社,1995)

7 Bent B K. A conceptual perspective on the comparison of object-oriented programming language. ACM SIGPLAN, 1996, 31(2):42~53

8 Helm R, Holland I M. Contract: specifying behavioral compositions in object oriented systems. Journal of the ACM, 1990,21(10):169~180

9 Xi Jian-qing, Wang Neng-bin. Using objects to represent and compute constraints. Journal of Software, 1996,7(1):59~64
(奚建清,王能斌.限制的对象表示和计算.软件学报,1996,7(1):59~64)

10 ILOG Server Version 2.1. Mountain View, CA94043, ILOG Inc, USA, 1996

11 Shao Wei-zhong, Yuan Shu-tao, Yang Fu-qing. The design and implementation of an object-oriented programming language case C++ in Jade Bird II System. Journal of Software, 1996,7(1):1~8
(邵维忠,袁曙涛,杨芙清.青鸟Ⅱ型系统面向对象语言 CASE C++的设计和实现.软件学报,1996,7(1):1~8)

12 Rumbaugh J. Object-Oriented Modeling and Design. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1991. 246~247

Modeling Object Relationships in Programming Language

WAN Jian-cheng ZHANG Shu-ming

(*Department of Computer Science and Technology Shandong University of Technology Ji'nan* 250061)

Abstract After a discussion of the importance of relationships among objects as a first-class modeling concept in object-oriented technology, a new object-oriented programming language, named as RCPP (relational C++), is presented and developed in this paper, which can directly and explicitly support the relationships among objects. Object relationships can be used to dynamically manipulate object behaviors and their propagation. In this paper, the authors explain the functional properties, the language model for the specification of object relations and its implementation mechanism, especially the semantic aspects of object relations. The running of RCPP programs is implemented first by translating RCPP source codes into C++ codes, and then by compiling and linking them to form executable programs.

Key words Object Oriented, object relationship, programming language, object-oriented modeling.