构件类和构件的概念及其定义语言和操作语言*

*顾 明 **仲萃豪

*(深圳高等职业技术学院计算机系 深圳 518055)

**(中国科学院软件研究所 北京 100080)

摘要 本文针对应用软件的特点,在类和对象概念的基础上,提出构件类和构件的概念,说明构件类和构件与 O-O 中类和对象概念上的异同. 为规范化描述构件类,给出了构件类定义语言和操作语言.

关键词 软件工程,应用软件,面向对象,定义语言,操作语言,构件.

中图法分类号 TP311.5

尽管 O-O 研究已取得不少成绩,但在应用软件开发中,O-O 方法并未得到普及应用,其中几个主要原因是:①O-O 对软件重用的支持主要通过继承机制,对用非 O-O 方法开发的软件,很难用 O-O 中的继承机制实现重用,因此,O-O 对应用软件中存在的大量已有的软件无法重用;②人们观念的改变需要一个逐步的过程.

为了克服以上 O-O 方法中的不足,目前国际上已开始了构件软件(Component Software)的研究,使软件象硬件一样能通过标准的组件来组装,从系统级和应用级上开始研究构件软件的规范化标准,近几年来国际上已出现了一些这样的标准,主要的有 IBM 和 APPLE 公司的 OpenDoc,Microsoft 公司的 OLE2 和 OMG 联合组提出的 CORBA.[1]

本文的构件类和构件的思想就是考虑到国际上构件软件标准的研究,并结合我们开发应用软件的实际经验所提出的.

本文提出的思想,已在一个实际的 MIS 项目中进行了应用,目前其中的一个子系统已在试点单位投入运行,受到用户的欢迎,并正在向其他两个应用单位推广使用.

1 构件类和构件的概念

构件和构件类概念的提出,主要是考虑到应用软件开发的实际需要,其次是受到面向对象中类和类实例概念的启发.

---

* 作者顾明,女,1962 年生,博士,讲师,主要研究领域为软件工程. 仲萃豪,1934 年生,研究员,博士导师,主要研究领域为软件工程.

本文通讯联系人:顾明,深圳 518055,深圳高等职业技术学院计算机系

本文 1996-10-17 收到修改稿

构件类是模板,它本身是一个静态的概念,但可以生成各种动态的构件.与O-O相比,构件类相当于类机制,构件就是由类生成的类实例——对象,在传统的程序设计中,构件相当于过程和其所使用数据的封装体.开发应用软件时,用构件类产生构件,通过构件的组装和控制来构造应用软件.我们基于类和类实例的思想,并对它们进行了扩充,把它们引入到设计阶段.

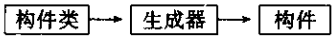

图1 构件类和构件的关系

这种扩充是从几个方面考虑的.第1,构件类和构件的概念不再仅仅限于编程阶段,作为一种设计思想,我们把它应用于设计阶段;第2,构件类和构件概念中可以有也可以没有继承机制,只要有封装即可;第3,通过聚集(Aggregation)和组装来支持重用;第4,通过构件操作界面(Interface)指针来实现构件之间的相互操作性;第5,构件类和构件之间的关系,不仅仅是类型和变量说明式的关系或调用创建函数和创建消息的关系,构件类可以通过一个生成器产生构件(见图1),而这种生成的方式可根据实际应用的需要分为几种情况.

(1) 变量类型和变量说明

构件类相似于变量类型,构件相似于变量,生成器的作用是从变量类型到变量之间的转换,例如,C++中类和类实例的关系就是这种情况.

(2) 函数生成

调用一个特定的生成函数,由构件类生成构件,例如,Smalltalk中的New函数,Eiffel中的Create函数,这时的生成器相当于函数调用的处理过程.[2]

(3) 宏替换和预处理

在构件类中可以有宏定义和预处理定义,对这些定义作宏替换或宏展开和预处理之后,就可以生成构件,这时的生成器就是一个作宏替换、宏展开和预处理的机制.在建立逻辑系统模型时,有些需要以后再加以详细说明的部分可以定义成宏替换和预处理.

(4) 文件描述

构件类也可以由一个文件的内容来描述,这时的生成器就是一个生成工具,通过生成工具,把由文件描述的构件类转换成构件.例如,在Informix-4GL中,可以通过屏幕表格说明文件来描述屏幕表格,然后,用屏幕表格生成工具生成最终的屏幕表格.

(5) 规则表达式

构件类中含有规则的描述,通过相应的生成工具生成构件.例如,UNIX中的Lex和Yacc就是由规则表达式生成程序的两个生成工具.

下面我们把构件类和构件类实例(构件)与O-O中的类和对象有什么异同做个比较.

(1) 在纯面向对象的设计中,类、封装和继承都是必不可少的.但对构件类而言,可以没有继承性,只要实现封装性即可,也就是说,构件类放宽了对继承性的要求.从这一点讲,构件类和构件不一定是O-O中的类和实例,而同基于对象(Object-Based)中的类和实例相似;

(2) 从对象和构件的生成方式上看,在O-O中,类和对象之间的实例化过程比较单一,而构件类和构件之间的实例化过程包含了以上所述的5种情况,这对应用软件的开发是很有实用价值的;

(3) 从概念上讲,OOP中的类和对象是编程时的概念,它们依赖于某种编程语言,而构件类和构件是设计时的概念,与具体编程语言无关,同一个构件类可由不同编程语言实现;

(4) 从对软件重用的支持上看,O-O 中对象对软件重用的支持通过继承机制来体现[3],而构件对软件重用的支持可通过两个方面体现:第 1,在构件概念中,虽然主要强调封装,可以没有继承,但并不排除继承;第 2,构件对软件重用的支持主要是通过聚集来体现,我们也把聚集称为组装控制,它的含义是开发可重用的构件库,定义构件之间的相互操作标准(Interoperation Standards),通过过程控制来组装生成应用程序;

(5) 从对象间和构件间的相互操作上,OOP 中对象间的相互操作通过公用接口来存取对象定义时的公用部分,这个公用部分可以是数据,也可以是操作函数,具体操作时,不同的 OOP 语言又有具体的规定;对构件而言,相互操作是通过构件操作界面指针进行的,不允许直接操作构件中的数据,只能通过指针操作指定的界面.

2　构件类定义语言

2.1 构件类的语法描述

抽象构件类语法如下:〈Comp_def〉::= Comp〈Comp_spec〉[〈Comp_body〉] END 〈Comp〉

由以上可见,一个构件类由构件类说明和构件类体两部分组成,其中构件类体可以为空,把构件类说明和构件类体分开的原因是考虑到同一个构件类说明可以由不同的程序设计语言来实现,即同一个构件类可以有多个不同语言实现的构件类体.

2.1.1　构件类说明的语法

〈Comp_spec〉 ::=	Spec〈Comp_name〉	构件类名
	[〈Comp_para_lists〉]	参数表(可以为空)
	〈Base_fields〉	构件类所属构件库名
	〈Operation_lists〉	操作界面表
	〈Relation_lists〉	关系表
	〈Attribute_lists〉	属性表
	END 〈Comp_spec〉	

由以上可见,构件类说明以关键字 Spec 开始,以关键字 END 结束,最多可以包含 6 个部分,凡是"[ ]"中的内容表示可以为空. 下面的符号"*"是 BNF 中的含义.

2.1.1.1　构件类名的语法

〈Comp_name〉::=〈Identifier〉

〈Identifier〉::=Letter{ Letter| Digital| '—' }*

构件类名由标识符标识,标识符是以字母打头的字母数字或下划线串.

2.1.1.2　参数表的语法

〈Comp_para_lists〉::=Para(〈Comp_para〉{,〈Comp_para〉}*)

〈Comp_para〉::=〈Identifier〉

参数表由多个参数组成,若参数表非空,则它至少包含 1 个参数,而参数可由标识符标识. 一个构件类的参数表可以为空,构件类带参数称为含参构件类,用实参替换形参,即可使含参构件类成为一般构件类.

2.1.1.3　构件类所属构件库名的语法

〈Base_fields〉::=Base〈Base_lists〉

〈Base_lists〉::=〈Base_name〉{,〈Base_name〉}*

构件类库名提供了一种将相关的构件类按照某种特征组织在一起的手段，这种特征可以是应用领域等，一个构件类也可属于多个构件类库，这样虽然带来信息冗余，但有时却可为实际应用提供方便.

2.1.1.4 操作界面的语法

⟨Operation_lists⟩::=Operation{⟨Operation_declaration⟩,}*

⟨Operation_declaration⟩::=⟨Operation_name⟩⟨Operation_para⟩[:⟨Return_values⟩]

以上说明，操作由操作名、操作参数和操作返回值3部分所组成. 操作界面是构件类对外部的唯一接口，界面实际上是一组操作函数指针，在构件类说明中，是对界面的一个定义，称为界面定义，是操作函数的一个说明，在构件类体中，是界面定义的具体编程实现，在构件类实例化时，界面定义不能实例化，只是为构件类体中的所有界面操作函数分配存储空间，并返回一个指向这些存储空间的界面函数指针表(见图2).

图2 构件操作界面的使用

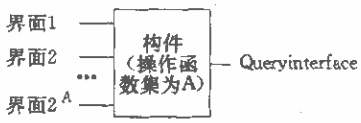

图3 构件及其操作界面

每个构件对不同的引用可提供不同的界面，因此，一个构件呈现在构件用户面前的就不只是一个界面，而是界面操作函数个数的幂集，即：设界面的操作函数集为 A，则该界面最多可有 2^A(A 的幂集)个不同的界面，系统中的每个构件都有一个构件操作界面查询(Queryinterface)的操作，通过该操作，用户可查询一个构件的界面情况，构件的界面如图3所示.

2.1.1.5 关系表的语法

⟨Relation_lists⟩ ::= Rela[⟨Inherit_R⟩]⟨Reference_R⟩

构件类之间的关系有两种：继承关系和引用关系，其中继承关系可以为空.

(1) 继承关系

⟨Inherit_R⟩::=Inherit⟨Supercomp_lists⟩ (1)

⟨Supercomp_lists⟩::={⟨Supercomp⟩;}* (2)

⟨Supercomp⟩::=⟨Comp_name⟩[⟨Rename_clause⟩][⟨Redefine_clause⟩]

⟨Rename_clause⟩::=Rename⟨Rename_part⟩{,⟨Rename_part⟩}* (3)

⟨Rename_part⟩::=⟨Old_name⟩As⟨New_name⟩

⟨Old_name⟩::=⟨Attribute_name⟩|⟨Operation_name⟩

⟨New_name⟩::=⟨Identifier⟩

⟨Redefine_clause⟩::=Redefine⟨Redefine_name⟩{,⟨Redefine_name⟩}* (4)

⟨Redefine_name⟩::=⟨Operation_name⟩

在上面继承关系的语法中，继承的父构件类是以 Inherit 开始说明的，由式(1)、(2)可以看出，允许多重继承，因而一个构件类可以有多个父构件类，子构件类继承所有父构件类的全部静态和动态特征.

由于允许子构件类有多个父构件类，因而名字冲突将是不可避免的，为此，提供了由关键字引导的换名子句，如式(3)所示，它将父构件类中的属性或名字按需要改成新名字.

有时子构件类还需要细化或增加父构件类的操作界面以满足自己的需要，为此，式(4)提供了重定义 Redefine 子句，指示将父构件类的某个操作界面重定义.

例如,假设"系统分析员(SA)"是一个构件类,而它的父构件类有"软件设计员(SOFT)"和"应用领域专家(AF)",则在 SA 的构件类说明中可能会有以下情况:

Inherit Soft Rename a1 As New_a1,op1 As New_op1,

Redefine op2;AF

上式说明,构件类 SA 继承了构件类 SOFT 和 AF 的属性和操作,并将构件类 SOFT 的属性 a1 换名为 New_a1,操作 op1 换名为 New_op1,并重新定义了操作 op2.

继承关系可以为空,即可以没有继承关系,在本文的以后内容中未做讨论,之所以在语法中给予描述,是为以后构件类的进一步研究留下扩充的接口.

(2) 引用关系

⟨Reference_R⟩::=Refe⟨Comp_name⟩⟨Interface⟩

⟨Comp_name⟩::=⟨Identifier⟩

⟨Interface⟩::= {⟨Operation_name⟩,}*

上式说明,一个构件类可以引用另一个构件类的操作界面,操作界面是操作界面语法描述中的操作名. 当用户引用操作界面时,先使用 Queryinterface 界面查询构件类界面的情况,然后再决定要使用哪个界面.

2.1.1.6　关系表的语法

⟨Relation_lists⟩::=Rela[⟨Inherit_R⟩]⟨Reference_R⟩

构件类之间的关系有两种:继承关系和引用关系,其中继承关系可以为空.

⟨Reference_R⟩::=Refe⟨Comp_name⟩⟨Interface⟩

⟨Interface⟩::= {⟨Operation_name⟩,}*

上式说明,一个构件类可以引用另一个构件类的操作界面,操作界面是操作界面语法描述中的操作名.

2.1.1.7　属性的语法

⟨Attribute_lists⟩::=Attr{⟨Attr_def⟩;}*

⟨Attr_def⟩::=In⟨Inside_attr⟩ Env⟨Env_clause⟩

⟨Inside_attr⟩::= {⟨Primitive_attr⟩;}* [⟨Create_time⟩][⟨Storage_priority⟩][⟨Modify_time⟩][⟨Use_frequency⟩]

⟨Primitive_attr⟩::=Interage | Real | Character |...

⟨Env_clause⟩::=⟨Version_spec⟩⟨Develop_env⟩⟨Run_env⟩⟨Readme_spec⟩⟨Documetation_name⟩

以上说明,属性由内部属性和环境属性两部分组成,其中内部属性包括预先定义的一些原始类,如整数、实数、字符等.

环境属性包括版本说明、构件类开发环境、构件运行环境、构件类自然语言理解和与该构件类相关的文档名称.

2.1.2　构件类体说明的语法

⟨Comp_body⟩::=Body⟨Comp_obj⟩⟨Comp_demo⟩ END ⟨Comp_body⟩

⟨Comp_obj⟩::=Obj[⟨Comp_para_lists⟩][⟨Local_para_lists⟩]⟨Body_implementation⟩

⟨Comp_demon⟩::=Demon⟨Comp_ins_name⟩⟨System_configuration⟩

以上说明,构件类体由两部分组成,一部分是构件类说明中操作界面的具体实现例程;另一部分为构件类的演示,由于构件类本身是不可执行的,所以要演示某个构件类的功能,

必须首先使之实例化,再加上必需的系统配置后方可实际运行.

3 构件操作语言COL

从构件类和构件的概念可知,构件有状态和生命周期,也就是说,构件是一个动态演变的实体,它可以生成、变化以至消亡,因此必须有一种机构来控制这种构件的动态性,我们把这种构件控制模拟成一系列的操作,即为以下主要的几类操作:(1) 构件的生成;(2) 构件的消亡;(3) 构件操作界面的查询;(4) 构件操作界面的生成;(5) 构件引用链的查询;(6) 构件操作界面的删除.

3.1 构件的生成

构件的生成是通过构件类的实例化来完成,关于实例化的过程,如前所述,有5种方式,对于这5种方式,定义以下3类操作:

(1) 对于第1种方式,生成实例的操作为:构件类名(实参) 实例名

因为构件类可能是含参构件类,在实例化时需进行形实参替换,实例名即为构件名.

(2) 对于第4种方式,生成实例的操作为:实例名 Create(构件类名(实参))

实例名即为构件名,Create为生成函数.

(3) 对于其它的3种方式,生成实例的操作为:实例名 构件类名(实参) 文件名

实例名即为构件名,在文件中可给出有关的宏替换和预处理说明,也可给出生成规则.

在一个具体的应用软件中,应该选择哪一种生成操作,这主要取决于所选择的具体编程语言,在系统实现时,COL需要一个解释器或编译器,为了使COL的解释或编译尽可能地简单和有效,对于不同的编程语言,尽可能采取与其语言本身类和实例对应关系中比较接近的生成方式,表1说明了几种常用编程语言和实例生成操作类型的对应关系.

表1 编程语言同构件实例生成的对应关系

具体编程语言	实例操作生成类型
C++	1
Smalltalk, Eiffel	2
Ada, 4Gl, C, Pascald等不支持类和对象机制的语言	3

3.2 构件的消亡

构件的消亡是指把某构件类实例删除,即该构件类减少了一个实例,形式为

Delete(构件类名,构件类实例名)

3.3 构件操作界面的查询

构件的操作界面查询是查询指定构件的全部操作函数,它返回一个操作函数的指针表,形式如下

Queryinterface(构件类名,构件类实例名,指针)

3.4 构件操作界面的生成

前面已经介绍过,同一个构件可以有多个操作界面,构件的用户可以通过Interface来形成自己所需的操作界面,在使用本操作前,应该先使用Queryinterface操作来得到构件的全部操作函数,从返回的操作函数指针表中再选择生成所需的操作界面,形式为

Interface(构件类名,构件名,指针)

3.5 构件引用链的查询

当某用户引用了某构件的操作界面时,该用户和该构件之间就建立了一个引用链,用户对构件操作界面的引用次数就是引用链数,用户每建立一个引用链,则链数就自动加 1,当构件未被引用时,链数为 0,只有链数为 0 的构件才可以被消亡.为了得到引用链数,可使用以下形式

Queryref(构件类名,构件名,整数)

其中整数是该操作的返回值,即引用链数.

3.6 构件操作界面的删除

操作界面的删除是指删除操作界面的引用,删除有两种:删除指定的操作界面和删除全部的操作界面,

Delete(构件类名,构件名,指针)　删除某构件指针所指的操作界面引用;

Delete(构件类名,构件名,ALL)　删除某构件所有的操作界面引用

4 结束语

为了有效地描述构件类和操作构件,我们在软件开发平台上开发相应的软件工具,包括构件类编辑器和构件操作控制器,目前这些工具已初具雏形,并正在完善之中.

参考文献

1 Brockschmidt K. Inside OLE2. Microsoft Press, 1994.

2 Coad P, Yourdon E. Object-oriented design. Yourdon Press, Englewood Cliffs, N. J., 1990.

3 James Rumbaugh, Michael Blaha, William Prewerlani. Object oriented modeling and design. Pretice Hall, 1991.

COMPONENT CLASS, COMPONENT AND THEIR DEFINITION LANGUAGE AND OPERATION LANGUAGE

*GU Ming　**ZHONG Cuihao

*(*Department of Computer Science　Shenzhen Polytechnic　Shenzhen*　518055)

**(*Institute of Software　The Chinese Academy of Sciences　Beijing*　100080)

Abstract　In this paper, component class and component concept are presented, the difference and consistence between them and class and object are compared. In order to describe component class and component standardly, component class definition language and operation language are given.

Key words　Software engineering, application software, object-oriented, definition language, operation language, component.

Class number　TP311.5