对象组装及其关联语义的自动维持*

万建成, 刘 嵩

(山东大学 计算机科学与技术学院,山东 济南 250061)
E-mail: wanjch@sdu.edu.cn
http://www.sdu.edu.cn

摘要: 讨论了对象组装概念及其对象关联语义的自动维持问题.在给出对象组装的模型之后,还对其语言描述、实现机制进行了阐述.它是基于专门用于组装的对象挂接,而不是常规对象接口.由此实现了面向问题域的对象间关联的动态建立和关联语义的自动维持,从而增加了对象建模的描述能力和使用的灵活性,降低了对象接口的实现和使用的复杂度.作为对象描述和对象作用的机制,它为软件 IC 的实现提出了新思想,其应用、规范和标准化将为软件设计的工业化生产提供新的设计技术.

关 键 词: 对象组装;语义自动维持;对象间关系;对象关联;软件重用
中图法分类号: TP311 文献标识码: A

对象关联或关系是面向对象软件技术研究的热点和重要发展,其目标是建立软件的面向对象模型,实现软件重用,降低复杂软件的设计难度.建立对象关联需要解决关联关系的描述和语义维持问题.处理对象关联的传统方法,是定义各对象对外界提供的操作界面(object interfaces),然后通过手工代码的设计,通过对象方法的复杂调用来控制对象的相互作用行为.在诸如 ER 扩展方法[1]、复杂关联[2]、OMT 模型[3]中的对象间关系的设计等对象间关系的研究,以及现行的软件设计环境中,都是以这种方法来处理对象关联的.由此造成系统中对象间关系模糊不清晰,复杂系统构造困难,难以实现对象间行为依赖关系的自动动态维持[4]等一系列问题.传统的程序语言中不能显式直接支持对象间行为关系设计,这是出现这些问题的根本原因.

本文讨论了对象组装概念及其描述和实现,研究了由此引发的对象关联语义的自动维持问题.它将复杂的关联语义用显式的方式描述出来,从而使对象间的操作关系的表达达到规范化、标准化和自动化.这对软件结构的清晰化、控制软件系统的复杂性具有十分重要的意义,也为软件重用思想的实现提出了新的概念和方法,是软件重用的高级阶段.

本文描述的对象组装是以用于组装的对象挂接接口(object hook/flange)而不是常规的对象接口(object interface)为对象建模和实现的描述依据.通过对象挂接接口的描述、实现和使用,可实现面向问题域的对象间关联语义的描述和维持,从而降低了对象接口的实现和使用难度.

相关的研究包括 Kristensen 等人[5]研究了基于角色(role based)的对象关联,但没有讨论对象的组装和关联语义的动态自动维持;Litman[4]研究了基于 Path-Based Rule 的相互依赖对象间关系的动态自动维持,它使用了时间戳(timestamps)来记录和激发关系的自动维持,并将维持的行为赋予类,因而实现的复杂度高且运行效率低.此项研究没有采用时间戳.

对象挂接和组装与现行的对象和行为的动态连接库概念不同.前者解决的是被关联对象和行为关系的动态建立和维持;后者仅局限于类和行为的运行时刻的动态替换.前者的实现是基于汇集并由此建立关联语义的动态自动维持的;诸如 CORBA,COM/DCOM,JavaBean 的后者是通过对象注册实现的.通过关联语义的动态自

* 收稿日期: 2000-04-28; 修改日期: 2001-02-27

作者简介: 万建成(1949－),男,江西进贤人,教授,主要研究领域为软件工程,面向对象技术,人工智能,自然语言处理;刘嵩(1975－),男,山东济南人,工程师,主要研究领域为面向对象技术.

动维持实现对象组装概念,是本研究的独特之处.

1 对象组装及其关联语义的自动维持

1.1 对象组装的概念

以面向对象的观点看,软件系统是为完成特定功能而建立起来的具有特定关联关系的对象集合.特定关联关系至少包括继承、友、聚集、汇集、使用、消息传递、事件激发、行为依赖等.关联分为松散关联、紧密关联.松散关联是可分离式的.

对象挂接(object hook/flange)是一种实现对象动态、直接、快速行为依赖连接的机制.这里的“连接”都是指在对象运行中而不是设计中的,加上发生挂接的对象的可变性,构成了挂接的动态性;所谓直接连接是指在挂接过程中不需要、也不可能获得任何额外代码的实现支持;由于存在着动态直接性,也就保证了挂接是瞬时完成并得以实现的.

对象挂接还具有以下一些特性:(1) 发生挂接的对象将自动维持连接和由于连接所引起的行为依赖关系;(2) 发生挂接的对象双方在挂接中充当不同的角色,它们分别是主动方和从动方.主动方是行为依赖的源对象,从动方是目标对象;(3) 对象挂接是数据驱动的,即:源对象有关数据的变化将引起目标对象的操作行为.这是对象挂接的语义所要求的.

对象组装是一种可分离式的对象关联,它是不同的对象为实现特定功能,通过对象挂接而建立起来的具有行为依赖关系的对象集合.由于对象挂接的性质,因而对象组装也具有构成上的动态、直接、快速性,以及行为的依赖性.这反映了对象组装的结构要求,也是建立对象组装的目的所在.

1.2 对象组装的关系模型

下面给出对象挂接和组装的关系模型.图 1 是原理模型,图 2 是实现模型.

在图 1 中,挂接的源对象 A 具有激发数据源 Position pos,目标对象 B 具有接收激发的数据和操作方法 TrigPos().源对象具有维持与目标对象关联关系的能力,通过它的方法 LinkObjBy(B)把目标对象 B 挂界到源对象上去.图中以箭头和环表示.

发生挂接的对象在挂接关系中充当了特定角色,为了表达角色的特性以处理复杂情况,可以为发生挂接的目标对象附加新的属性和操作方法.在图 1 中,这就是目标的 typeOfAttA AttA 和 typeOfAttM AttM().

当把多个 B 对象挂接到 A 对象上,A 就拥有了一个 B 对象的汇集(collection).另外,所有以 B 为基类导出的类的对象,都可以与源对象发生挂接关系.这可以通过在类 B 的继承链上使用虚方法作为被激发方法而得以实现.这样,被挂接的目标对象就形成了源对象的一个异构汇集.这些就是图 2 所说明的内容,其中 OBJ 代表 B,ObjList<OBJ>是 A 的成员.

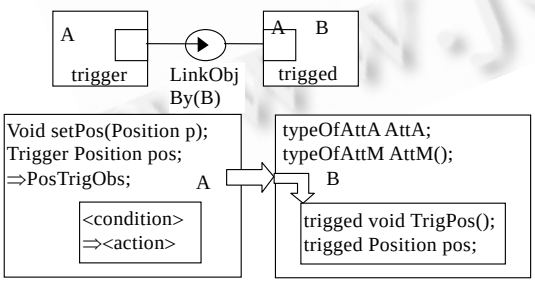

Fig.1 Model of object assembling
图 1 对象组装的原理模型

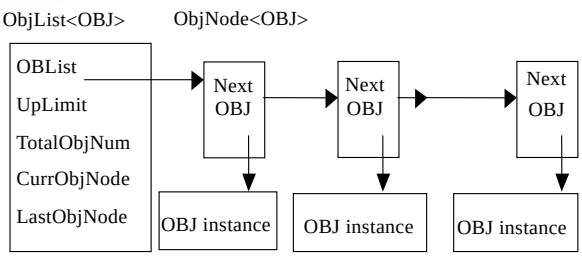

Fig.2 Working mechanism of object assembling
图 2 对象挂接的运行机制模型

更复杂的,一个源对象可以拥有多个挂接对象的汇集(collection),每个汇集表达一种挂接关系;甚至,一种挂接关系的源对象可以成为另一种挂接关系的目标对象;进一步地,处于同一种挂接关系的目标对象还可以将挂

接关系反转过来,从而成为反转挂接关系的源对象.如果把挂接看成是一种信息通道,那么存在反转的挂接关系就被看成是一种具有双向行为依赖的双向传送信息通道.

可见,对象组装是一个具有复杂挂接关系和语义的对象的集合,它是由对象关联[2](object association)和关联语义的自动维持而形成的,因而不同于一般的对象关联概念.

2 对象组装的描述

由以上说明可以看到,实现对象组装的条件是:建立并维持规范化、标准化和确立固定的连接方式.这包括挂接关系源对象激发数据的确认和控制和实施依赖关系操作方法的规定(可以使用策略).这一切都由对象挂接的描述所规定.

对象挂接的描述包括以下内容:(1) 类型:属性关联,标识关系的类型;(2) 方向:确定挂接关系的主动方、从动方的对象类型;(3) 约束:关联的基数,可建立 0:*n*,1:*n*,但不能建立 *n*:1 关系;(4) 行为依赖:建立源对象与目标对象属性之间依赖的关系不变式(invariant),主动方参与依赖关系的数据成员,其值的变化将引起依赖关系的检查;主动方激发依赖关系操作的条件,当该条件成立时激发维持依赖的操作;依赖关系被激发后的操作.

对象挂接和组装所建立的对象间关系,包括关联数据的更新、变化,变化后状态/值的检查,检查成功后依赖行为的操纵,都应该根据挂接关系的语义自动地受到监督和执行.

3 对象挂接组装的实现

本文思想的实现方法是:使用类似 C++的文法形式,设计对象挂接的描述文法;以 C++为目标语言,设计实现了对象挂接描述的程序转换生成器;使用 C++环境实现了对象挂接的运行.所以,下面将以与 C++类似的文法描述,说明对象组装的描述和实现方法.

3.1 对象挂接的定义

```
class A {                                  //对象挂接的源对象定义
public: void setPos(Position p);           //更新数据成员 pos 的方法,需要附加对 pos 状态的检查
private: trigger Position pos⇒classOfPosTrigObs PosTrigObs;};
                                           //trigger 是新关键字,pos 被定义为源对象参与挂接的数据对象,
                                           //PosTrigObs 用以保持参与挂接的目标对象表
A::PosTrigObs {                            //源对象激发依赖关系操作的条件和维持依赖的操作
   <condition>                             //<condition>和<action>都可以包含对 A 和 PosTrigObs 中
⇒<action>      };                          //目标结点属性、挂接的附加属性和方法的检查和方法调用
void A::setPos(Position p) {               //经转换后的 setPos()方法
                                           // setPos()的原始操作
                                           //转换器追加的激发条件的检测,调用 A::PosTrigObs 方法}.
```

通过调用规范的方法把作为目标的 B 对象挂接到对象 A 上,具体做法是,A.PosTrigObs.LinkObjBy(B&b, <paramData>,<paramMethod>).在此,PosTrigObs 的公开使用可以区分可能存在的多种挂接关系,对于复杂系统设计这是必要的;虽然这样会部分地破坏对象的封装特性.

3.2 实现对象挂接的基础类模板

这是维持挂接的目标对象表的基础模板 ObjList 类.它提供了维持对象间一对多关系必须的控制状态和操作方法.前者包括关联对象个数的上、下界,关联对象的个数,表头、表尾和当前正在访问的对象;后者包括表的初始构造和清除,对象的插入、删除、检索、移动,表状态的获得.整个表的操作过程维持了对象关系的完整性.类的主要定义如下:

```
template <class OBJ> class ObjList
{
```

```
public: ObjList(WORD LowL=0, WORD UpL=defaultUpLimit) :…{…};
        ObjList(OBJ * obj, WORD LowL=0, WORD UpL=defaultUpLimit):…{…};
        bool LinkObjBy(OBJ * obj) {…};          //在 UpLimit 的限制下挂接一个目标对象
        void removeLinkAt(WORD objn) {…};       //从对象表中移走第 n 个对象
        OBJ * getNext() {…};                    //从对象表中获得下一个对象
        OBJ * getFirst() {…};                   //获取对象表的第一个对象
        WORD getTotalObj() {…};                 //读取当前对象表中对象的个数
        WORD getUpLimit() {…};                  //读取对象表标的上界 UpLimit
private: OBJ * OList;                           //对象表
        WORD LowLimit, UpLimit;                 //对象表的下界和上界
        WORD TotalObj;                          //当前对象表中对象的个数
}.
```

4 几点说明和处理

组装的继承性:由于存在类的继承性,挂接关系定义的被挂接类以及由该类导出的类的对象,都可以被挂接,并维持挂接的行为依赖关系.同理,使用虚方法作为行为关联语义的实现,例如上例中的 TrigPos()方法,可以很方便地改变导出类的语义实现.另外,由于挂接的源类中挂接接口的公共(public)特性,使得源类的导出类对象依然具有该挂接接口的能力.

附加属性和附加方法:被挂接的类 B 是作为扩展类 AttachedB 的成员出现的,附加属性和附加方法也是它的成员,所以,通过类 AttachedB 的对象可以方便地实现附加属性和附加方法的访问.附加方法可以包含对附加属性和 B 类对象的访问和处理.

对象的双向组装:上述对象挂接所建立的行为依赖关系是单向的,称为对象的单向组装.如果发生挂接的对象可以互为源和目标对象,就形成了对象的双向组装.

5 结束语

本文提出的思想,除双向组装外,已经通过程序转换器的方法以 C++为目标语言得到了实现,并获得了实验性的应用.进一步的工作包括:(1) 实现并研究双向组装;(2) 使被挂接对象的操作方法可以通过参数,甚至通过操作策略(strategy)[6]的选择加以控制,也可以将策略用于依赖操作调用的条件检查上,以提高对象挂接的行为的动态可控能力.

致谢 山东大学计算机科学与技术学院的卢雷、赵合计副教授、鹿旭东讲师在论文的形成中给予了热心支持,提出了很多有益的建议;张署明、倪惠青同志对本文的试验工作给予了大力帮助,在此一并表示感谢.

References:

[1] Saiedian, H. An evaluation of extended entity-relationship model. Information and Software Technology, 1997,39(7):449~462.

[2] Kristensen, B.B. Complex associations: abstractions in object-oriented modeling. In: Proceedings of the 9th Annual Conference on Object-Oriented Systems, Languages & Applications(OOPSLA'94). Portland, Oregon: ACM, 1994. 272~286.

[3] Rumbaugh, J. Models for design: generating code for association. Journal of Object-Oriented Programming Model & Design, 1996, 8(2):13~17.

[4] Litman, D., Patel-Schneider, P.F. Mishra, A. Modeling dynamic collections of inter-dependent objects using path-based rules. In: Proceedings of the 12th Annual Conference on Object-Oriented Systems, Languages & Applications (OOPSLA'97). New York: ACM, 1997. 77~92.

[5] VanHilst, M., Notkin, D. Using C++ templates to implement role-based designs. In: Futatsugi, K., Matsuoka, S., eds. Proceedings of the 2nd JSSST International Symposium on Object Technologies for Advanced Software. Berlin: Springer-Verlag, 1996. 22~37.

[6] Gamma, E., Johnson, R., Booch, G. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Longman, 1994.

Object Assembling and Automatic Maintenance of Its Association Semantics*

WAN Jian-cheng, LIU Song

(*School of Computer Science and Technology, Shandong University, Ji'nan* 250061, *China*)

E-mail: wanjch@sdu.edu.cn

http://www.sdu.edu.cn

Abstract: In this paper, the concept of object assembling and automatic maintenance of its association semantics are discussed. After presenting the model of object assembling, the mechanism of its language description and implementation are described, which is based on object hook interfaces, rather than the ordinary object interfaces, and through which the domain-oriented dynamic manipulation of object association and automatic maintenance of association semantics are achieved. This has greatly enhanced the description ability and the flexibility, and decreased the complexity of object association. As a mechanism of objects and object relation description and manipulation, it gives a new idea to realize software IC, and its usage, formalization and standardization will provide a new technique for industrialized production of the software.

Key words: object assembling; automatic maintenance of semantics; object relationships; object association; software reuse

* Received April 28, 2000; accepted February 27, 2001

国际未来软件技术学术会议(ISFST 2002)
征文通知

由华中科技大学、日本软件技术协会、联合国大学澳门软件研究中心主办，华中科技大学承办的国际未来软件技术学术会议将于 2002 年 10 月 23 日~25 日在武汉召开。

会议征文范围:

软件开发方法/工程技术、软件处理(建模/管理)/能力成熟度模型(CMM)、Web 相关技术、数据库/数据管理、软件体系结构与构件、测试与验证、系统演进和维护、计算机应用中人的方面、紧密集成软件、软件质量/质量保证、基于统计的质量和过程管理、约束/关键链路理论、信息安全、电子商务、电子政务与电子教育、实时、智能与移动计算、软件集成环境数字企业与企业软件等.

重要时间:

论文全文或 Extended 摘要截止期: 2002 年 6 月 30 日

论文录用通知: 2002 年 8 月 15 日

修改后的论文全文截止期: 2002 年 9 月 30 日

联系方式:

彭世卿 430074 湖北省武汉市 华中科技大学科学技术协会

E-mail: hxwu@public.wh.hb.cn Fax: 027−87547971 Tel: 027−87543051