计算机科学1998 Vol 25 No. 4

面向对象可视化重用构件的语言动态特性研究*)

On Programming Language's Dynamic Characteristics in Implementing Object-Oriented Visual Reuse Part

桑大勇 刘西洋 蔡希尧 陈 平 彭 亮

(西安电子科技大学软件工程研究所 西安710071)

摘 要 By developing reusable Chinese Reports Feature under IBM Visual Age for Smalltalk, the authors have proposed a composite visual reusable part model into which meta representation is introduced. In this paper, dynamic characteristics of the programming language and (or) SDE (software developing environment) needed for implementing the part model are discussed. Furthermore, implementation possibility and strategy in 3 main-stream OOPLs, Ada95, C++ and Java are presented, with emphasis on that of Java.

关键词 Meta representation, Composite visual reusable part, Dynamic event trigger, Runtime type information

1 引言

可视化重用构件是可视化程序设计的基础,目前主流的可重用构件模型(CORBA, OLE /ActiveX, JavaBeans[2]等)都不太适合可视化重用构件模型[1],作者在完成 IBM VisualAge for Smalltalk[3](以下简称 VA Smalltalk)环境下的可重用中文报表构件时建造的可视化重用构件模型具有以下特点:

·全面支持可视化程序设计,满足 MVC 模型[4]并加以适当扩充;

·支持复合构件与子构件间的动态组装及相应的语义约束,用户可以在程序运行时动态改变复合构件的组装结构(如用户对子构件进行拖放操作改变子构件的隶属关系),使得对可视化编程的支持程度更高。

该构件模型现已用 Smalltalk 实现,能够支持跨平台可视化环境 VA Smalltalk 和 IBM-VisualAge Generator 下开发应用系统时的重用。如何将这种模型推广到其它几种主流的 OO 语言环境,是需要进一步深入研究的课题[1]。本文从构件模型的上述特点入手,讨论模型实现语言所需要的动态特性,进而研究几种主流的面向对象语言实现模型的可能性及其实现方案。

2 可视化重用构件模型的特点

2.1 构件模型对 MVC 模型的扩充

该模型以 MVC 为基础,同时引入对构件生命周期的管理及对象网络协调的机制。在 MVC 模型中,可视化构件由模型(Model)、视图(View)和控制器(Controller)组成,模型描述构件中有关应用逻辑的属性和行为;视图描述与模型的属性和行为相依赖的构件侧面(典型情况是构件的可视化部分);而控制器是联接并协调模型与视图的中介。我们提出的可视化重用构件模型也符合 MVC 模型,每个构件是由一组分工明确又相互协同的对象组成的对象网络,图1给出了一个构件模型实例的 UML[5]描述。此图例示的是当构件属性发生变化时,对象网络的协同工作过程。图1中的 aField 对应于 MVC 中的模型;aVisualPart 代表构件的图形显示部分,对应于 MVC 中的视图;anInputPolicy 和 aVisualPolicy 对应于 MVC 中的控制器,前者控制重用构件的输入事件对模型的影响,后者控制构件的绘制方式(如模型属性发生变化后构件可视部分的重画等);aPartBuilder 和 anEditPart 是模型扩充的部分,前者负责构件生命周期的开始,进行重用构件的创建(即对象网络的实例化),后者负责对象网络中各对象之间工

*)本文得到九五军事预研课题"软件重用及可重用部件库"(15.1.4.2)的支持。桑大勇 博士生,刘西洋 博士生,蔡希尧 教授、博士导师,陈 平 教授、博士导师,研究方向为面向对象软件工程,彭 亮 硕士生。

作的协调。这种构件模型实际上是一种多设计样本(Design Pattern)协同的模型。

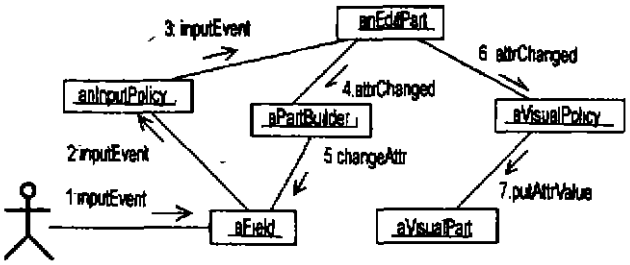

图1 可视化重用构件的对象网络模型实例

2.2 复合构件模型

复合构件的动态组装结构是用户通过可视化操作定制重用构件的基础。为了支持复合构件的动态组装结构，构件模型必须满足以下要求：

(1)父/子构件间的双向引用可达性。构件属性的动态继承是可视化复合重用构件模型实现的关键技术之一[5]，属性动态继承机制要求不仅父构件可以访问到子构件，子构件也要能动态访问到父构件。

(2)子构件的类型约束。组装结构变化时为了保持子构件属性的一致性，父构件各对象用以记录组装结构的 components 属性中存放的是子构件中的对象而不是对象引用[5]。由于父构件要以统一的方式控制子构件的行为，因而子构件要满足统一的类型属性，Smalltalk 之类弱类型化或无类型语言本身就能支持，而对于强类型化的语言来说，就要求子构件的各个对象能分别统一到相应的基类之上，这给构件种类的扩充带来一定的影响。

(3)子构件的应用语义约束。从应用领域的角度看，并非所有种类的构件都可以作为某类复合构件的子构件，如将报表中的页码构件放置在表体构件(父构件)中是不允许的，这就要求每个构件本身能够对其父构件的合法性加以约束。

3 模型实现需要的动态特性

基于可视化重用构件模型所具有的上述特点，为了实现这种构件模型，程序设计语言及开发环境必须具备以下动态特性。

(1)动态触发参数化事件的能力。由于复合构件支持动态组装结构，使得子构件无法在编译时确定，当上下文使用统一的方式(发出消息)操纵子构件时，消息的接收对象本身(子构件)会动态发生变动，因而除了必须将消息的接收者参数化以外，更需要将消息及其参数一起动态确定，这就要求开发环境支持这种参数化的事件触发机制。另外由于我们的构件模型是一个对象网络，这些对象视不同的构件有可能不同，但它们之间为了协同工作，需要使用统一的事件触发机制，因而也需要事件响应者的数据化。这样，语言环境必须能够实现如下所示的 API：

signalEvent (receiver, selector, arguments);

(2)运行时获取类或类型完整信息的能力。当用户在编辑时使用某个构件的时候，我们必须创建该构件对象网络中的每个对象，这时必须能够获取每个对象的类的完整信息，诸如其构造函数(Constructor)等。一个对象接收到自己的类型或类中没有定义的消息时，如果语言环境是强类型化的(如 C++，Ada95以及 Java 等)，程序在编译时就不能通过；如果语言环境是弱类型化或无类型的(如 Smalltalk)，程序在运行时将发生异常。为了避免这类异常的发生，当我们将对象作为参数向其发送消息时，只有获取对象的类的全部接口定义信息后才能确定消息能否被接收对象理解。

从子构件的应用语义角度来看，对父构件的合法性验证主要是对其类型进行验证，这也要求实现语言具有动态获取对象的类信息的能力。这种能力是实现动态对象的基础，因而也是可视化重用构件的前提。

4 主流 OO 语言实现模型的可能性或方案

不同语言中动态类型信息获取能力有所不同，可视化重用构件模型实现的手段及难易程度也有所差异。由于 Smalltalk 语言具有比较完善的类结构自描述体系，用 Smalltalk 实现可视化重用构件非常理想。下面我们主要讨论 Ada95、C++以及 Java 这几种语言实现模型时的可能性，并给出关键部分的 Java 解决方案。

4.1 Ada95和 C++中模型实现的可能性

Ada95支持使用 s'class 预定义属性[6]获取类的标识，C++也提供了 RTTI(运行时类型鉴别)机制[7]，但程序运行时都只能获取类型标识，不能获取以类对象之类的元对象存储的完整类型信息。为了实现可视化重用构件模型，关键在于要人为地增加很多额外的数据结构用以记录对象的元信息(类中定义的属性、函数接口以及它们的存取特性等)，模拟元对象机制。

Frank-Buschmann 等人[10]给出了一种能够用 Ada95或 C++这样没有元对象机制的 OO 语言实现

的、基于对象反射的设计样本。依据这种样本，实现时我们可以将构件的结构划分成两个层次：元层次和基础层次。元层次由元对象(即前文所述的额外数据结构)组成，提供构件对自身结构和行为的描述，每当我们定义一个新的类时，使用元对象协议操纵相应的元对象。基础层次实现应用逻辑，易变部分由元对象封装，保持了相对独立性。可视化重用构件动态获取类型信息的操作相当普遍，动态改变类型信息(如运行时增加新的方法接口)的操作相对较少，这使得元对象协议相对简单一些。

4.2 Java 中模型的实现方案

JDK1.1实现了受限的对象反射机制，使我们有可能比较容易地动态获取类的信息。下面简单地给出前述重用模型用 Java 的实现方案。

(1)Java 运行时类或类型完整信息的动态获取。JDK1.1实现了一组核心反射的类[9]，这些类包括 Class、Method、Member、Field、Constructor 和 Modifier 等，任何对象通过 getClass()方法可以获取其类型对应的类对象，通过类对象就可以获取类定义中所有成员变量、方法、构造函数以及它们的存取特性(public，private 等)，这些信息分别存储在上面几个类的实例中。Java 的反射机制之所以是受限的，是因为有些类的实例只能由虚拟机创建，用户程序不能创建实例，如 Field，Method 和 Constructor 类。

(2)构件类型约束的实现。从 Object 派生六个类 CommonField，CommonInputPolicy，CommonEditPart，CommonVisualPart，CommonVisualPolicy 和 CommonPartBuilder 分别存放构件模型中六个对象的共同协议，每类新的原子构件的六个类继承这六个基类。CommonField 定义5个类方法 inputPolicyClass，editPartClass，visualPartClass，visualPolicyClass 和 partBuilderClass，具体类型构件的 field 类重置这些方法，分别返回一个类对象代表构件网络中某个对象的类。例如

```
public Class inputPolicy Class()
{
  //假设该构件的 inputPolicy 对象的类是 CommonInputPolicy
  Class ans;
  ans=Class.for Name("CommonInputPolicy");
  return(ans);
}
```

另外复合构件的六个类也直接继承上述六个基类，并增加一个 components[]属性存放子构件中对应的对象，子构件即 components 元素的类型分别为对应的基类。例如

```
class Composite Field extends public CommonField
{
  public components=new CommonField[];
};
```

(3)子构件应用语义约束的实现。为了实现子构件应用语义的约束，子构件的 field 类增加一个方法 acceptedParentParts()，用以返回该子构件可接受的父构件种类(由合法父构件种类的 field 类对象所组成的数组)。当用户将一个子构件(field 对象为 childField)添加或拖放到一个复合构件(field 对象为 ParentField)中时，可通过调用下面的算法验证是否满足应用语义约束。

```
public Boolean accepted (CommonField childField, CompositeField parentField)
{
  Class[] acceptedParentFieldClasses;
  acceptedParentFieldClasses = childField. acceptedParentParts();
  for (int i = 0; i < Array. getLength (acceptedParentFieldClasses);i++)
    if (acceptedParentFieldClasses[i]. is Instance(parentField))
      return(true);
  return(false);
}
```

(4)动态事件触发机制的实现。当我们将消息的接收者连同消息名、消息参数一起作为参数触发事件时，首先要获取消息接收者对应的类对象，进而获取类中定义或继承来的全部方法对象，再一一同消息名及消息参数进行匹配。下面给出一种算法。

```
import java.lang.reflect.*;
  public void signalEvent(Object receiver, String selectorName, Object[]parameters)
{
  Method methods;//receiver 类型的所有方法
  Boolean isValidSelector=false;//某个方法是否与所给方法匹配
  methods=receiver.getClass().getMethods();
    for(int i=0;i<Array.getLength(methods);i++){
      Class[]pars=methods[i].getParameterTypes();
      int pars_no=Array.getLength(pars);
      if(!methods[i].getName().equals(selectorName)||
        pars_no!=Array.getLength(parameters))continue;//参数个数不匹配
      isValidSelector=true;
      for(int j=0;j<pars_no;j++){
        if(!pars[j].isInstance(parameters[j]){
        //parameters[j]不能转换为 pars[j]类型的值，因而方法不匹配
        isValidSelector=false;break;
        }
      }
    if(isValidSelector){
      //methods[i]与参数中给出的方法选择子匹配，激活该方法
    methods[i].invoke(receiver,parameters);
    break;
    }
  }
  if(!isValidSelector){
  //receiver 不能执行所给的方法，进行例外(Exeption)处理……
  }
}
```

结论 构件自身类型描述的元数据，是文中可视化重用构件的基础[1]，因而 OO 语言动态类型完整信息的获取是使用该语言实现重用构件的关键技

计算机科学1998 Vol. 25 №. 4

人际通信中图的作用及其对人机对话的意义

Pictures in Human-Human Dialogue and Its Implications to Human-Computer Dialogue

刘正捷
（海事大学计算机系　大连116026）

朱宗元
（国家科委西南信息中心　重庆400013）

摘　要　The difficulty for the construction of picture semantics oriented human-computer dialogue comes mainly from the complexity in human's understanding of picture semantics and the utility of the semantics in dialogues. The paper proposes a layered model for picture semantics and discusses the classification for pointing actions to picture semantics, attempting to provide a conceptual framework for analyzing certain types of human-human dialogue. The discussion on this basis about the human-computer dialogue suggests that (1) choices should be made for layers and dimensions in the picture semantics model according to the need of user's task and be limited to as shallow layers as possible, (2) the layers chosen determine the types of pointing action and (3) introduction of the reference context could facilitate the handling of pointing actions.

关键词　*Human-computer dialogue, Picture, Semantics, Pointing action*

目前人机界面之所以不能支持针对图的语义的对话，原因在于机器既不理解图的语义，也缺乏对图的语义的运用能力。相比之下，面对图的对话者形式的人际通信有两个基本要素，一是对话双方对图的语义的相同理解，这构成了双方通信的公共基础(common ground)[2]；二是双方在表达与理解信息时对图的语义有相同的运用方式。因此，在建立针对图的语义的人机对话时，必须使机器中图的语义模型以人对图的语义理解为基础，使人机对话符合人在对话中对图的语义的运用方式。然而，人际对话中对图的语义理解和运用现象的复杂性正是建造此类系统的难点所在。为此，必须认真分析图的性质以及图在人际通信中的使用，特别是人对图的语义理解和语义运用，从中获得有关建立相应人机对话的有益启示。

本文首先从人际通信角度对图的概念作出限

术。另外为了支持复合重用构件的动态组装结构，语言要能够实现动态事件触发机制，这也要求能够运行时获取类型信息，本文给出的 Java 实现方案已经在 IBM VisualAge for Java 上原型实现，得到了初步的验证。

致谢　本文的研究工作得到了国防科工委、电科院和 IBM 中国公司的支持，作者在此深表感谢。

参 考 文 献

[1]刘西洋等，中文报表可视化复合重用构件的研究与开发，西安电子科技大学软件工程研究所研究报告，1997年7月

[2]Java Beans 1.00-A. SUN Microsystems Corp., Dec. 1996

[3]VisualAge for Smalltalk Programmer's Guide to Building Parts for Fun and Profit(Third Ed.), IBM Publication SC34-4496, 1997

[4]Krasner G.E. et al., A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80, J. of Object Oriented Programming, 1(3) 1988

[5]Martin Fowler and Kendall Scott, UML Distilled—Applying the Standard Object Modelling Language, Addison Wesley Longman, 1997

[6]John Barnes, Programming in Ada95, Addision-Wesley, 1996

[7]ANSI/ISO Working Paper for Draft Proposed International Standard for Information Systems—Programming Language C++, ANSI X3J16/96-0018, ISO W921/N0836, Jan. 1996

[8]Frank Buschmann et al., Pattern-Oriented Software Architecture—A System of Patterns, John Wiley & Sons, 1996

[9]Java™ Core Reflection, SUN Microsystems Corp., 1996