计算机科学1999Vol.26№.1

可视编程环境模型*)

Visual Programming Environment Model

王鳌东　沙一鸣　尤晋元

(上海交通大学计算机系　上海 200030)

Abstract This paper defines a layer model of visual programming environment, including components layer, system structure layer and representation layer. Components layer describes the internal implementation of the components. System structure layer is to describe the structure relationship, management relationship and user define relationship of the components designed by users. Representation layer will represent the three relationships and create the destination program according to these.

Keywords Visual Programming Environment, Visual Programming Environment Model(VPEM), Component

当前的可视编程环境正在将所见即所得作为基本环境设计原则，并以基于消息处理的编程方式作为用户进行开发的主要方式。在这些基本原则下，出现了各类不同的可视编程环境，大致可以分为三类：

1)以资源方式来刻画用户界面，这类可视编程环境的编程和界面构造相互独立，其代表有 Visual C++, Visual J++[1], Java Workshop[2], Symanter Cafe 等。这些工具是给专业人员使用的，具有很大的灵活性，但开发工作量仍然比较大。

2)以用户界面构造为核心，用户将各种控制的源代码直接嵌入各个控件中。这类工具以事件、方法、属性来刻画控件，将搭建成最终的应用系统，其典型代表是 Visual Basic[3], Power Builder, Delphi 等，这种方式的灵活性比第一种方式弱，但开发工作量大为减少，对于开发各种规模的界面型应用程序具有特别显著的优点，适合于业余和专业人员使用，

3)以输入和输出来刻画控件。这类可视编程环境通过连接构件之间的输入输出方式完成各种流程控制，不需要开发人员写任何源代码，其典型代表是 Java Studio[4]。这种方式的灵活性差，一般只能应用于较小的应用实例。由于开发者只要了解构件的输入输出接口，而并不需要了解某种具体的语言，所以较适用于非专业人员开发较小的程序模块，

虽然，这三种可视编程环境表面上不同，但是本质上都有统一的内部结构。本文下面定义了一种统一的模型，用以刻画可视编程环境的内部结构，我们将此称为 VPEM(Visual Programing Environment Model)模型。

1. 可视编程环境模型

1.1 模型概述

VPEM 模型是一个三层结构(图1)，其最底层是构件层，负责描述可视编程环境中的基本构件功能。中间层是系统结构层，用于刻画用户系统的结构，说明各个构件之间的关系，表现层是最上面的一层，用于实现编辑和运行状态的显示、调试等功能。在构件层和系统结构层之间是构件接口描述，用于刻画各个构件以及构件管理的接口。在系统结构层和表现层之间的接口是程序生成方式，实际上就是系统结构向目标程序的映射方式。

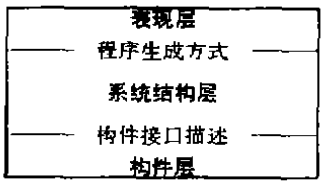

图1　VPEM 的层次结构

1.2 构件层以及构件接口描述

构件层刻画可视编程环境中的基本元素：构件。

*)　本文工作得到"九五"国家重点科技项目(96-737-01-05)资助，王鳌东　博士生，主要研究方向为分布对象计算，尤晋元　教授，博士导师，国务院学位委员会计算机评议组成员。

当前所有的可视编程环境，都是面向构件的，尽管在不同系统内的名称可能不同。在 Java Workshop 和 Java Studio 中以 Java Bean[5] 作为构件，而 Visual Basic 中以 VBX 控件或 OCX 控件作为构件。

首先要区分构件、构件类型和构件接口定义方式这三个概念。构件是用户开发系统的基本元素，例如某个窗口或某个按钮；而构件类型与构件的关系实际上是类和对象的关系，每一个构件一定是某个构件类型的实例。窗口、按钮和文本框等本身就是构件类型；各种构件类型都有其构件接口，构件接口定义方式是将各个构件的接口统一定义的方式。比如，一种构件接口定义方式将构件类型看作是一个由名称、执行程序、一组属性、一组事件和一组方法所构成的五元组。如果将构件接口定义方式看作是类，则各种构件类型则是该类的各种对象实例。所以构件、构件类型和构件接口定义方式之间的关系如图2表示。

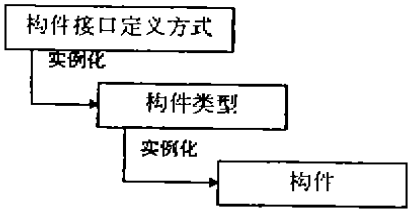

图2 构件、构件类型和构件接口定义方式之间的关系

构件内部层和系统结构层之间的构件接口描述的核心是一组构件的接口，而构件接口的刻画方法就是一个构件接口定义方式。也即每个构件层的实现都对应一个构件接口定义方式。所以，要讨论构件接口描述，首先要规定一个构件接口定义方式。以上面的五元组构件接口定义方式为例，构件接口描述中就需要有：

```
get_name
get_program                    获取程序
get_property_list              获取属性列表
get_event_list 获取名称        获取事件列表
get_method_list                获取方法列表
```

作为向上层提供服务的基础，构件层还需要管理各种不同的构件类型，所以在构件接口描述中，还包括有管理不同构件的接口，例如下面的一组接口：

```
get_component_type_list
    获取构件类型表
insert_component_type component_type
    加入一个新的构件类型
remove_component_type component_type
    删除一个旧的构件类型
```

当然，这个接口并不唯一，作为构件内部层的构件类型管理部分，还可以实现构件类型分组管理，查询等。

不同的构件接口定义方式可以产生不同的构件接口描述，所以构件接口定义方式是本层次实现的基本依据，由构件接口定义方式产生构件接口描述，同时通过两次实例化，得到构件内部层中的实现。

当前已有的各种可视编程环境中，构件接口定义方式大都比较类似，所以构件接口描述也较相似。构件层实现的主要区别是如何获取构件的接口，其可能的实现方式有固定型、配置文件型和程序型。具有配置文件型和程序型开放式接口的构件层，其构件的实现与其运行所依赖的系统无关，只需要实现一个简单的构件类型管理和构件接口获取程序，这样可以将复杂问题分解，减轻可视编程环境本身的开发工作量。

无论构件的接口怎么定义，可视编程环境都应当在界面上提供编辑构件接口各个部分的能力。

1.3 系统结构层

系统结构层是可视编程环境模式的中间层，其作用是用某种数据结构刻画用户需要开发的系统。一般情况下，经过用户开发后的系统由一组经过个性化处理的构件组成，比如对构件的属性赋予了特定的值等等。所以系统结构层需要解决的问题是处理构件之间的关系和刻画构件的个性化。

系统结构层的主要目的是刻画构件之间的关系。构件之间可能有多种关系。不同的关系，刻画的方式也不完全相同。在系统结构层实现的构件关系有：结构关系、管理关系和用户关系。

结构关系用于说明构件之间在显示界面上的相互关系，一般是包含关系。例如窗口和窗口中的按钮关系。一般情况下，每个构件都有一个包含它的父亲。有一个特殊的构件（如桌面），包含最顶层的构件。一般情况下，构件之间的结构关系实际上是一棵树。可视编程环境必须提供某种机制，让用户可以刻画和了解各个构件之间的关系。图3描述了一个典型的结构关系，

管理关系实际上是对构件之间消息传送的管理。现代的图形环境都使用消息处理的机制来开发。整个系统可以收到来自环境的一些消息，例如某键被按下，鼠标发生了移动或时间过了一秒钟等。这些消息应当如何被处理呢？各个图形环境采用的方式不尽相同。而可视编程环境一般都由自己来管理这些消息的传递。假设用户在某个窗口内定义了一个按钮。当这个按钮收到了点击的消息时，一般都首先

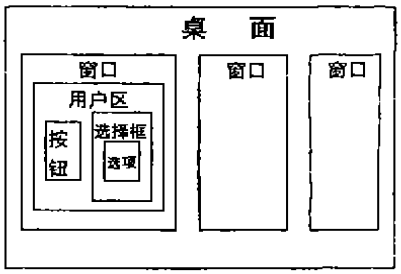

图3 构件之间的树型结构关系

寻找这个按钮的点击消息处理程序，如果用户没有定义这个按钮的点击消息处理程序，那么应当如何处理呢？常见的有四种情况：

1）抛弃该消息 系统不对接受到的消息作任何处理。

2）转给系统统一的消息处理程序处理 系统规定了一种两层管理关系，根是桌面，而其它所有用户定义的构件都是叶节点。叶节点不能处理的消息很自然地由桌面进行统一处理。

3）将消息转发给包含这个按钮的窗口或其他构件包含者 这样的管理关系同构件之间的结构关系是一致的。

4）将消息转给自己的管理构件[6] 这种管理关系是最灵活的。这时需要在构件之间再定义一种管理关系，当一个构件不能处理某个消息时，自动将消息转发给其管理构件。一个构件的管理者可能不止一个，此时就需要定义管理的优先级。构件在不能处理消息时，应当首先将消息发给优先级最高的管理构件，如果它不能处理，则发给优先级次之的，以此类推。

用户关系是用户开发的系统中规定的各个构件之间的关系，一般不是直接针对构件的，而是构件的消息、方法或属性等之间的关系。例如，用户系统中可能规定当用户在某个文本框按下回车，则向某个按钮发出点击消息。这个关系是文本框的键盘消息和按钮的点击消息之间的关系。用户开发系统，实际上主要的工作就是刻画构件间的用户关系。可视编程环境必须提供让用户刻画用户关系的机制，常见的有程序刻画法和图形刻画法。

程序刻划法可以刻画各种复杂情况，适合于给比较专业的程序员使用。图形刻画法使用简单，不需要编写代码，但是刻画能力较差，不能用于开发较大的系统，适用于初学者。

根据上面的描述，为了刻画用户开发的系统，可视编程界面还应当提供分别用于描述结构关系、管理关系和用户关系的三张图。当前实际使用的可视编程环境一般还只能提供其中一部分。

1.4 表现层以及程序生成方式

表现层是整个模型的最顶层，其主要功能是将用户系统表现出来，这里所说的"表现出来"有三部分含义：① 在编辑时提供所见即所得的效果；②在运行时提供正确的结果；③提供调试的能力。这三部分含义实际上对应于构件的三种状态。即编辑态、运行态和调试态，每个构件都应当有这三种状态，而这些状态的区别仅仅是它们的消息处理方式不同。运行态的消息处理程序实际上就是用户编写的消息处理程序或相当于消息处理程序的部分，调试态与运行态消息处理的区别是在运行态的消息处理程序的基础上，加上系统自动加入的调试消息。而编辑态的消息是由可视编程环境处理的这些消息被用于处理构件接口中规定的允许修改的接口，例如初始属性、消息处理程序等。

编辑态的表现是由各个构件自己决定的，不是表现层中需要解决的主要问题，而主要解决的问题是如何根据构件以及上面提到的构件之间的关系，来生成用于运行态和调试态的最终代码。这也就是程序生成方式需要解决的问题。

程序生成方式中，要处理的主要问题是如何根据构件之间的结构、管理和用户关系，完成最终程序的生成。对于运行态程序，需要注意到生成程序的结果是某个宿主系统的代码，可能是直接的一个窗口系统的 API，也可能是某种高级程序，需要进一步编译后才能运行，还可能是一种自定义的伪代码，供解释运行。这些情况虽然生成的代码不同，但是其生成的方式都是相同的。

构件间的结构关系，说明的是构件互相之间的包含，这对于最后的程序生成会有两个方面的影响：1）构件显示时的包含与覆盖关系。现有的窗口系统大都对构件的包含有直接的支持，可以在用户系统启动时，通过调用宿主系统的一些 API 或类似过程来实现。2）构件的名称表示。是指如何通过一个标识来唯一指定一个构件。一般情况下，可以以构件的结构关系为基础，采用层次型命名方式来唯一标识一个构件。

构件间管理关系的处理方式主要依赖于宿主系统的消息传递模式，当宿主系统的消息传递模式与管理关系相同时，可以直接利用宿主系统的消息传递模式。如果两者不同，那么就需要进行底层处理，

利用宿主系统更底层的一些消息，来控制消息传递的流程，达到对上面透明的效果。现有的窗口系统大都提供这样实现的可能性。

构件间的用户关系是用户程序的主体。根据用户关系生成最终程序，依赖于用户关系的定义方式。现有的情况有：

1)程序刻画用户关系且刻画程序与最终程序的代码相同。这是最简单的一种情况，只需要将用户写的刻画程序放置于一个适当的生成程序的位置即可。

2)程序刻画用户关系，但刻画程序与最终程序的代码不同。这种情况由于代码不同，所以需要对刻画程序进行编译，转化为最终程序的代码，再放置于适当的位置。

3)图形刻画用户关系。这种情况下，需要根据图形生成相应的代码。例如进行某个消息传递、属性修改或方法呼叫等。这些代码一般都是非常直接和简单的。生成的代码可以和第一种情况相类似地进行处理。

以上对三种构件关系的处理，配合上一定的启动和结束代码，可以完成最终程序的生成。

对于调试态程序，系统需要将控制权在特定的时刻交给调试器，由其决定下一步的工作流程。这些特定的时刻，可以是在每次收到消息时、完成消息处理时或者每一步程序处理时。所以当生成调试代码时，需要对管理关系和用户关系的生成进行一定的修改，使其可以实时地向调试器发出一定的请求调试消息。除此以外，调试态程序和运行态程序没有什么本质区别。

综上所述，表现层完成的主要工作是最终程序的生成，而生成过程，实际上就是对结构、管理和用户关系的一次映射，直接将这些关系映射为最终代码。开发可视编程环境的重要一步就是定义出这种映射关系。

2. VPEM 与现有可视编程环境的关系

VPEM 是一个抽象的可视编程环境模型，与现有的诸多可视编程环境有着直接的关系。我们以 VPEM 的眼光，来查看一下现有的一些典型可视编程环境。Visual Basic、Visual C++ 和 Java Studio 分别是我们提到的三种可视编程环境的典型代表。而 VJPE(Visual Java Program Environment)则是我们自行开发的 Java 编程环境。

Visual Basic

构件内部层：
构件接口定义方式：以消息、属性和方法来刻画
构件来源：外部来源，使用 VBX 或 OCX 接口定义
构件接口获取方式：程序型
系统结构层：
结构关系：直接支持
管理关系：无(抛弃未处理消息)
用户关系：程序刻画型
表现层：
生成代码：内部代码
结构关系生成：与 Windows 系统对应
管理关系生成：无
用户关系生成：进行编译

Visual C++

构件内部层：
构件接口定义方式：以消息和方法来刻画
构件来源：外部来源，使用 OCX、ActiveX 等，有少数内部来源
构件接口获取方式：外部来源使用程序型，内部来源使用固定型
系统结构层：
结构关系：通过对象包含表现
管理关系：隐含对应于结构关系
用户关系：程序刻画型
表现层：
生成代码：C++源代码
结构关系生成：与 MFC 类库相对应
管理关系生成：利用 MFC 类库中的消息传递机制
用户关系生成：直接插入生成程序

Java Studio

构件内部层：
构件接口定义方式：以消息、属性和方法来刻画
构件来源：外部来源，使用 Javabean
构件接口获取方式：程序型
系统结构层：
结构关系：Java 的 AWT 中体现
管理关系：与结构关系对应
用户关系：图形刻画型
表现层：
生成代码：Java 源代码
结构关系生成：与 AWT 对应
管理关系生成：利用 Java 的内部支持
用户关系生成：生成图形对应的源程序

VJPE

构件内部层：
构件接口定义方式：以消息、属性和方法来刻画
构件来源：外部来源
构件接口获取方式：配置文件型
系统结构层：
结构关系：直接支持
管理关系：无(抛弃未处理消息)
用户关系：程序刻画型
表现层：
生成代码：Java 源代码
结构关系生成：与 AWT 对应
管理关系生成：无
用户关系生成：直接插入源程序

由上可知，VPEM 可以归纳现有的主要可视编程环境，但是也有一些功能，在现有的环境中都还没有支持，例如复杂的管理关系等。

3. VPEM 的主要特点

VPEM 的主要特点是：①使用构件内部、系统结构和表现层这样的层次模型来刻画可视编程环境的各个部分。②使用构件接口定义方式、构件接口

(下转第68页)

知识描述和集成应用提供了一个途径。

实现这种知识组织模型的理想方法是援例推理(CBR)技术。援例推理系统可以通过回忆以前求解问题的案例，重用案例的解(必要时需进行案例调整)来求解新的问题，并且通过保存求解新问题的案例使系统具有学习能力。

援例推理方法在知识管理方面有较大的优势，案例将问题描述、求解方法、应用背景等结合在一起，通过案例可以体现具体问题求解过程中的知识，包括不易表达的缄默知识。CBR 可以用于知识管理的各个方面，如表示、存储、检索、学习等，并且可以方便地与数据库系统和信息系统集成。一个案例可以认为是数据，可由数据库系统对案例进行存储和管理，由用户对案例进行解释。对于一个用户来说，放在计算机案例库中的先前案例是信息，可以通过信息技术对案例进行组织和处理，在问题求解过程中，案例也可以通过援例推理方法作为知识使用。这意味着案例可以从数据、信息和知识三个角度来理解，可以很容易地将案例集成于数据库系统、信息系统和知识系统中。

结束语 企业知识作为企业的一种重要的无形资产正被人们所接受，由此便产生了对企业知识管理的需求。企业知识管理的目标就是对分布于企业内的知识有效地组织、分配和利用，最大限度地实现企业知识的共享和协作。实现这一目标的有效途径是建立企业知识库系统，其核心任务是建立企业知识库。理想的知识系统应提供关键字信息检索、超文本浏览、超媒体知识库和模拟与动态推理要求。

CBR 技术在支持企业知识管理方面有一定的优势，它可以用于知识管理的各个方面，如表示、存储、检索、学习等，并且可以方便地与数据库系统和信息系统集成，然而 CBR 在许多方面有待于进一步的研究和发展，如案例的匹配、适应性调整等。如果将 CBR 技术与其它技术和方法结合，如与特定的 AI 问题求解方法和认知模型结合，必将在知识管理方面发挥更大的作用。

参 考 文 献

1 Suhn B, et al. Enterprise decision support using Intranet technology. Decision Support System, 1997, 20

2 Grant R M. Toward a knowledge-based theory of the firm. Strategic Management Journal, 1996, 17

3 Tsoukas H The firm as a distributed knowledge system: a constructionist approach. Strategic Management Journal, 1996, 17

4 Barr J M, Magaldi R V. Corporate Knowledge Management for the Millennium. In: Smith I, Faltings B eds. Advances in Case-Based Reasoning Third European Workshop/EWCBR-96. Springer-Verlag, 1996. 487～496

5 Greenes R A, et al. Knowledge management as a decision support Method: a diagnostic workup strategy application. Computer and Biomedical Research, 1989, 22

6 Spender J C. Making knowledge the basis of a dynamic theory of the firm. Strategic Management Journal, 1996, 17

(上接第78页)

定义和构件之间的关系，将构件的接口问题一般化。③使用结构、管理和用户这三种关系作为编辑和程序生成的基本要素。

VPEM 模型是一个比较通用的模型，指明了可视编程环境的结构，以及各个部分需要解决的问题。现有的主要可视编程环境都可以归结为这个模型的一个特例。同时，本模型也提出了新一代可视编程环境应当加强的功能，例如提供更丰富的构件接口定义、加强管理关系的刻画等。

由于可视编程环境模型的探讨还在起步阶段，我们将会在一些新的可视编程环境的开发中，验证和发展本模型。

参 考 文 献

1 Microsoft Developer Network. Microsoft, July 1997

2 Weaver L, Jervis B. Inside Java WorkShop. 1996

3 Davis H. Visual Basic 5 Secrets. IDG Books Worldwide, Inc 1997

4 Java Studio Home Page. Available at: URL: http. // www. sun. com/studio/cover. html

5 Java Beans(TM) API specification version 1.01. Graham Hamilton(Editor), Sun Microsystems, July 24, 1997

6 Java 1.1.4 Specification. Available at: URL. http // java. sun. com

7 Visual Programmung Languages Can Compete with Traditional Programming Languages. Available at. URL: http: // www orst. edu/Dept/research/asti/pubhc-html/connect/con72/visual. html