计算机科学1998 Vol. 25 №. 4

# 引入元表示的可视化复合重用构件模型研究*)

On Composite Visual Reusable Part Model with Meta Representation Introduced

刘西洋　桑大勇　蔡希尧　陈　平

(西安电子科技大学软件工程研究所　西安710071)

**摘　要**　Considering the limitations of the meta representation in CORBA, OLE/COM and Java Beans, a composite visual resuable part model with meta representation is proposed based on the development of a reusable chinese reports part under IBM VisualAge for Smalltalk. The essential technologies related to its implementation and its implementation strategies in various object-oriented programming languages and systems are also disscussed.

**关键词**　Meta representation, Composite visual reusable part model, Multi-coordination system collaboration, Multi-Design pattern collaboration

## 1　引言

可视化程序设计以面向对象范型为基础，以可重用软件构件为基本构造单元，正越来越多地被用于关键(Misson critical)应用软件系统的设计与实现，而不是仅仅局限于人机界面设计。研究实践表明：上述关键应用软件系统的功能和非功能进化，迫切要求软件技术支持运行时的递增式有序软件更新[1]。目前工业界广泛基于的一阶对象技术(以C++为代表)显然难以满足上述要求。与此同时，以对象反射(Reflection)、元数据以及元对象协议为代表的经典动态对象技术已经提供了现实的对象系统自适应机制，"动态对象技术被专门设计成构造频繁演化的软件系统，使得软件系统演化的代价正比于其中必要的变化部分的规模，而不是正比于整个软件系统的规模"[1]，因而关键应用软件系统的开发迫切需要引入经典动态对象技术。在可视化程序设计的前提条件下，上述动态对象技术的引入将直接落实到面向可视化程序设计的可重用构件模型——可视化重用构件模型之上，因为可视化重用构件模型是可视化程序设计得以实施的基础和前提。还应看到，经典动态对象技术的精髓在于对象系统的高阶抽象，即将对象系统划分为关于目标应用领域的对象层系以及关于对象系统自身的元对象层系，对象层系中反映不同应用语义的不同对象，在元对象层系中被统一地加以表示和操纵，其中关于对象以及对象系统自身结构和行为的运行时表示即元表示，是元对象层系得以建立的基础，因而在可视化重用构件模型中引入经典动态对象技术的一个首要前提是：引入关于可重用构件自身的元表示。

进一步分析可以看出，可视化重用构件模型本身就需要引入元表示，否则可视化程序设计环境将无法一致，也无法有效地获取其所操纵的可重用构件的接口定义，从而难以在可重用构件之间建立可视化的语义关联，可视化程序设计因此将难以实施。事实上，中立于体系结构、操作系统、程序设计语言以及软件开发环境的统一元表示，是任何工业化的

*)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。刘西洋　在职博士生。桑大勇　博士生。蔡希尧　教授、博士导师。陈　平　教授、博士导师，主要研究方向为面向对象的软件工程。

[3] James Rumbaugh, Object-Oriented Modeling Technology, Tutorial, OOPSLA'94

[4] Erich Gamma et al., Design Patterns: Elements of Reusable Object-Oriented Software, 1994

[5] W. Zimmer, Relationships Between Design Patterns, Proc. of PLoP'94

[6] R. Eisenhauer et al., Architecture-Handbook for Software Architecture, Internal report, 1994

[7] Frank Buschmann, et al., Pattern-Oriented Software Architecture, 1996

[8] Analysis Patterns: Reused Object Models, Addison Wesley Longman, Inc., 1996

[9] K. Beck, Smalltalk Best Practice Patterns, Volume 1: Coding, Prentice Hall

[10] J. O. Coplien, Advanced C++-Programming Styles and Idioms, Addson-Wesley, Reading, MA, 1992

可重用构件模型必须引入的基础设施[2]。正因为如此，当前工业界主流的可重用构件模型无一例外地引入了关于可重用构件的元表示，其中 CORBA (Common Object Request Broker Architecture)引入了接口定义语言 IDL 以及接口仓库 IR[2]，OLE/ActiveX 引入了与 CORBA 类似的对象描述语言 ODL 以及类型库[2]，JavaBeans 则引入了关于 Java Bean 的元表示 BeansInfo。

但是应当看到，由于受到与之相应的可重用构件模型的制约(因为可重用构件模型从根本上决定了与之相应的元表示机制)，因而 CORBA、OLE/COM 以及 JavaBeans 事实上已经引入的这些元表示机制普遍存在着如下所述的局限性：

(1)至始至终从一个而不是多个角度建模可重用构件，因而无法将全方位的错综复杂的构件整体元表示简化为若干集中于构件某一侧面因而相对简单独立的构件局部元表示；

(2)着重描述单个可重用构件所定义的属性、事件和服务，对构件关联的描述仅限于与构件设计实现相对应的继承结构和静态组装结构，对构件系统中必不可少的动态组装结构和语义约束缺乏显式的描述，从整体上表现为单纯强调可重用构件个体而忽视可重用构件系统；

(3)此外，CORBA 中的 IDL 和 IR 几乎没有考虑可视化程序设计。OLE/COM 初步有所考虑。JavaBeans 虽然在设计伊始就将支持可视化程序设计作为其可选功能，并且目前已经推出相应的可视化程序设计环境，如 IBM VisualAge for Java，但与基于 Smalltalk 语言的可视化程序设计环境 ParcPlace-Digitalk VisualWorks 以及 IBM VisualAge for Smalltalk(以下简称 VA Smalltalk)相比，由于受到 JDK 类库(尤其是其中的 AWT)设计的制约，JavaBeans 对可视化程序设计的支持还相当初步。因而如何将基于 Smalltalk 的可视化程序设计环境中已经相当成熟和先进的概念、模型和方法有效地应用于基于 JavaBeans 的环境中，是当前迫切需要研究的问题。下文将对此作进一步的论述。

VA Smalltalk 环境下的中文报表构件(以下简称中文报表构件)，是西安电子科技大学软件工程研究所在 IBM 中国公司软件中心的支持下开发的可视化中文风格商务报表构件。其中构件的涵义在于：基于构件技术开发的中文报表软件，整体上又以可视化复合构件(而非完整应用)的形式存在，进而有效地被最终应用所重用。这正是中文报表构件与 Excel 中文版等现有中文报表软件的显著差别。来自商务报表的领域需求以及 VA Smalltalk 环境的技术需求，最终要求构造中文报表构件的可重用构件模型以及相应的元表示机制必须满足以下条件：

(1)全面支持可视化程序设计，可重用构件应能被显式地划分为支持可视化程序设计的编辑时[1](edit-time)部分和支持最终中文报表应用的运行时(run-time)部分；

(2)以经典 MVC(Model-View-Controller)模型[3]为起点，从多个角度出发建模应用领域中的可重用构件，如中文商务报表中的文本、迭代器(支持数据源为动态集合因而无法静态确定行数的报表)和预览器构件，从而有效地简化上述构件的设计、实现和维护；

(3)显式地引入并实现构件系统的动态组装结构和局部/全局语义约束机制。在中文报表中，其对应的应用场景(Scenario)为，中文报表构件整体上作为复合构件在可视化构件组装编辑工具中由用户定制。

然而，现实情况是，CORBA、OLE/COM 目前并不满足上述条件[2]，VA Smalltalk 迄今为止还不支持 JavaBeans。VisualAge 系列本身所定义的构件模型 Parts[4,5]。尽管已经提供了坚实的可视化程序设计支持，但还远远不能满足条件(2)(3)，因而在 Parts 的基础上进一步扩充 MVC 模型就成为一种自然而且合理的选择，扩充的结果就是下文论述的可视化复合重用构件模型。

## 2 可视化复合重用构件模型

如前所述，VisualAge 系列定义了自己的可视化构件模型 Parts。关于 Parts 的详尽描述已在[4,5]中给出，这里需要进一步强调的是：

(1)整个 Parts 建立在派生于 MVC 模型的设计样本(Design Pattern)Observer[6]之上，因而能够有效地支持构件以及构件系统之间的复杂事件依赖关系；

(2)与 OLE/COM、OpenDoc[2]以及 JavaBeans 相类似，在 Parts 中构件的接口定义为：(属性集，方法集，事件集)。不同的是，Parts 严格定义了构件与构件之间属性、方法和事件的语义关联，以及这些语义关联和构件的图形化注记，而这正是可视化构件模型的一个基本特征；

(3)从 Parts 模型到其实现，明显地存在着与[6]

1)在 JavaBeans 规范中被称之为设计时(design-time)。2)拒绝采纳 CORBA、OLE/COM 的依据并不是 VA Smalltalk 环境的限制，事实上，VA Smalltalk 既支持 CORBA，也支持 OLE2。

中给出的许多设计样本之间的严格对应关系，Parts模型及其实现从某种意义上甚至可以被看作为一个多设计样本集成系统的范例[3]。Parts的这一特色理应在扩充之后的可视化复合重用构件模型中得到更充分的体现。关于Parts的上述理解正是下文提出可视化复合重用构件模型的基本前提和依据。

### 2.1 基于多参照系协同的构件模型

从不同角度出发全方位地考察同一问题，进而"分而治之"，是人们认识和解决复杂问题的普遍方法，软件系统的开发当然也不例外。从基于UML(Unified Modelling Language)[7]的"4+1"View，ODP(Open Distributed Processing)中的ViewPoint，到AOP(Aspect-Oriented Programming，URL：http://www.parc.xerox.com/spl/projects/aop)中的Aspect，无一不是上述普遍方法在软件开发中的具体体现，区别仅在于："4+1"View侧重于通用信息系统的OO分析与设计，ViewPoint着眼于开放分布系统的建模，二者实际上都定位在分析设计层次，其形式化描述往往是不可执行的，而AOP完全定位在程序设计层次，Aspect及其集成的形式描述表现为可执行的AOP源代码，这些源代码经AOP引入的编译器Aspectweaver自动转换为可执行的目标代码。不难看出，三者之间事实上存在着很强的互补性。但到目前为止，有关的研究工作还在各自独立地进行，因而如何将三者有机地统一起来是亟待进一步研究的问题。

具体分析可视化重用构件面向的设计可以看出，尽管MVC模型已经从模型、视图和控制器的角度出发建模逻辑上的同一构件，并且控制器又可以进一步划分为输入控制器和输出控制器，但是仅仅这样划分还是远远不够的，不足之处主要在于：

(1)没有充分考虑构件的生命周期管理。逻辑上的一个构件必然要实现为至少包括模型、视图和控制器对象的对象网络，并且还要考虑构件系统的局部/全局语义约束，因而其创建和撤销已经远不止是OO语言中的类实例生成和撤销；

(2)没有充分考虑构件的编辑定制。对逻辑构件的任何可视化编辑定制最终都要落实到与之相应的对象网络，乃至子构件和父构件的协调变化上，否则无法满足应用定义的构件系统局部/全局语义约束，而MVC中的控制器仅仅支持视图与模型之间的一致性约束。在MVC模型的基础上针对这些不足进行相应的扩充，就得到了基于多参照系协同的可视化重用构件模型〈businessObject，visualPart，inputPolicy，visualPolicy，partBuilder，editPart〉。图1给出了该模型结构的UML类图描述，其中PartBuilder着眼于构件的生命周期管理，EditPart着眼于构件的编辑定制，其设计依据分别来自[6]中的设计样本Builder和Mediator。并且该模型的每一部分都符合Parts模型的定义，都可以被看作是Parts模型中的构件，Parts模型由此而被集成在整个模型中。文[8]中进一步给出了由图1所定义的构件对象网络的一个协同工作实例。

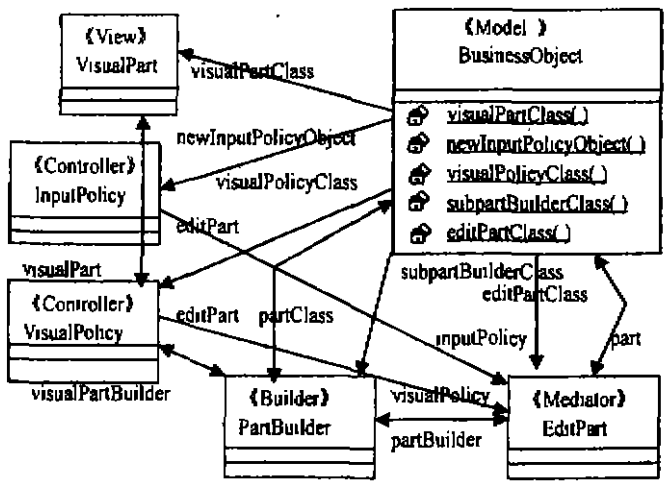


图1 基于多参照系协同的构件模型

由图中可以看出，设计样本MVC、Builder以及Mediator已经被显式地引入到模型中，下文还将进一步引入Composite(参见2.2)以及Chains of Responsibility，Parts模型的多设计样本集成特色由此得到更充分的体现。但是还应当看到，这里所得到的构件模型还难以支持构件系统的动态组装结构和局部/全局语义约束机制，因而需要进一步进行扩充，扩充之后就得到建模构件系统的复合构件模型。

### 2.2 建模构件系统的复合构件模型

假设构件$Child_1,\cdots,Child_n$是复合构件Parent的子构件，则对于$Child_i(1\leq i\leq n)$对象网络中的任意对象$O_j^{Child\,i}(1\leq j\leq 6)$，$O_j^{Child\,i}$满足：

$O_j^{Child\,i}\in O_j^{Parent}.components$，$O_j^{Child\,i}.parentPart=O_j^{Parent}$

3)考虑到[6]的主要作者同样来自IBM这一事实，这一点就不难理解。

即 Parent 对象网络中与 $O_i^{Child'}$ 相对应的对象 $O_i^{Parent}$ 借助其集合属性 components 引用并管理 $O_j^{Child\,j}$，同时 $O_j^{Child'}$ 借助其属性 parentPart 反向引用 $O_i^{Parent}$。由此不难推出：$O_i^{Parent}$ 与 $O_j^{Child\,j}$ 实际上构成了一棵多叉树 $Tree_j(1\leq i\leq n, 1\leq j\leq 6)$，由 $Tree_j$ 构成的森林实现了复合构件的对象网络与其子构件对象网络之间的网络互联（并且这种互联是双向的和引用可达的），最终构成复合构件的网际网结构。

复合构件的组装结构可以用 Z 语言形式地描述为：

[businessObject visualPart inputPolicy visualPolicy partBuilder editPart]
part :: = ⟨aspect1: businessObject, aspect 2: visualPart, aspect 3: inputPolicy, aspect 4: visualPolicy, aspect 5: partBuilder, aspect6: editPart⟩ parent, child: part

∃ i child. aspect$_i$ ∈ parent. aspect$_i$. components
⇒∀ i child. aspect$_i$ ∈ parent. aspect$_i$. components $(1\leq i\leq 6)$

### 2.3 可视化复合重用构件模型的元表示

针对模型的层次结构，相应地将其元表示划分为以下三个层次：

（1）图1所示的构件各个组成部分的元表示。如前所述，构件的各个组成部分都可以被看作是符合 Parts 模型的构件，因而其元表示可以落实到 Parts 已有的元表示机制。

（2）图1所示的构件对象网络的元表示。由 BusinessObject 类维护构件其它组成部分的类信息并定义获取这些类信息的类方法（即图1中 BusinessObject 类图中描述的类方法）。同时 InputPolicy、VisualPolicy、PartBuilder 和 EditPart 的实例也维护相应的网络引用信息，如 visualPart、editPart 和 partBuilder 等等。

（3）复合构件的元表示。集中体现在2.2节所描述的复合构件结构的网际网表示。

## 3 实现中涉及的关键技术

**1）动态获取类的完整定义（而不仅仅是类标识）。**Smalltalk 语言关于类的元协议直接就能做到这一点，OLE/COM、CORBA 以及 JavaBeans 同样如此，但是同样的结论并不适用于 C++、Ada95以及类似的 OO 语言和系统[8]。

**2）消息选择符作为参数的对象消息传递机制。**Smalltalk 同样直接支持这一机制。这一机制在 OLE/COM 对应于 IDispatch 接口，在 CORBA 中对应于动态调用界面（Dynamic Invocation Interface），在 Java 中尽管没有直接的支持，但是可以利用 Java 反射接口实现等价的机制[6]。

**3）显式地将构件的行为（对应于构件各个组成部分的类的方法）划分为编辑时部分和运行时部分。**在 Smalltalk 环境中借助方法分组机制就可以做到这一点，但是类似的机制在其它 OO 语言和系统中并不多见。

**4）构件在可视化编辑时的实例化。**构件在可视化编辑时的实例化过程实质上就是构件对象网络的创建过程，图2给出了这一过程基于 UML 协同工作图的描述。

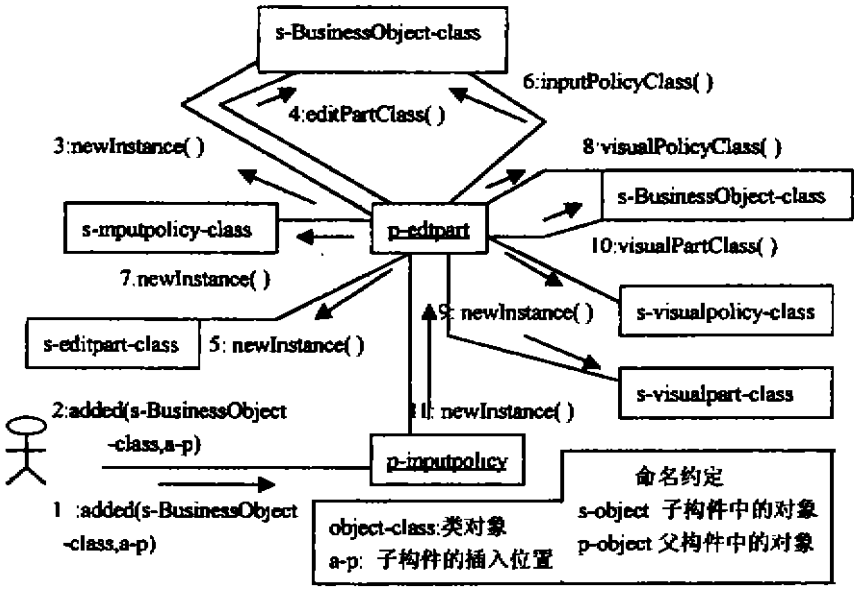


图2 构件在可视化编辑时的实例化过程

**5)复合构件中的类型约束和应用协议约束。**[8]已经对此进行了详细的论述，这里不再赘述。

**6)构件属性的动态继承(即复合构件中子构件对父构件属性值的共享)。**之所以称之为继承是因为这种值共享关系非常类似于类之间的单重继承关系。子构件共享父构件属性 attribute 值的算法可以用 Smalltalk 语法描述为：

```
attribute
  attribute isNil ifTrue:[self parentPart notNil[attribute:
    =self parentPart attribute]]. ^attribute
```

不难看出，这种实现实质上对应于[6]中的设计样本 Chains of Responsibility。

## 4 可视化复合重用构件模型的各种实现策略

尽管可视化复合重用构件模型来源于 VA Smalltalk 环境，但这并不意味着这种模型仅仅局限于 Smalltalk，基于其它 OO 语言和系统同样有可能实现这种模型，实现的可能性和难易程度关键取决于这种 OO 语言或系统的动态特性。

对于 CLOS 和 SOM(System Object Model)[5]这样的动态 OO 语言和系统，完全可以借助其反射机制实现构件的元表示，进而实现整个可视化复合重用构件模型；对于 C++和 Ada95这样的 OO 语言，由于其不提供运行时的完整类定义(仅能获取类标识)，可以采用类似于CORBA 中 IDL、IDL 预编译器以及 IR 的机制，动态提供类的完整定义。同时也可以考虑采用[9]中提出的设计样本 Reflection；对于 OLE/COM、CORBA 这样的构件模型，由于已经与 Parts 模型层次相近，但还不支持可视化程序设计，因而主要应当定义构件以及构件语义关联的图形化注记及其语义，并实现相应的构件组装编辑器；对于 JavaBeans，由于与 Parts 模型完全相似，并且已经支持可视化程序设计，因而主要应当依据2.1、2.2以及2.3的论述进一步实现整个模型。文[8]对此进行了详细的论述。

**结论** 本文提出的可视化复合重用构件模型通过引入三层元表示，增强了构件的重用定制能力以及跨平台能力。基于上述模型开发的面向中日韩市场的可重用中文报表构件，目前已经可以在跨平台[4]可视化应用开发环境 IBM VisualAge for Smalltalk 以及 IBM VisualAge Generator 下被应用开发所重用，其β版已交付 IBM 中国公司软件中心测试。进一步的深入研究将着重考虑以下问题：

·设计样本的组装能力以及 OO 语言的动态特性对这种能力的影响；

·建立可视化复合重用构件模型的形式化描述，其必要性正如[10]中所述："软件重用只有在引入形式化方法的条件下才可能最终获得成功"。这里引入形式化描述的关键在于形式化表达2.1所述的多参照系及其协同，而这完全可以参考 ODP 中体系结构语义的形式化模型。

**致谢** 本文的研究工作得到了国防科工委和电科院预研项目的支持，作者在此深表感谢。

## 参考文献

[1] Laddaga R. et al., Dynamic Object Technology, CACM, 40(5)1997

[2] Orfali R. et al., The Essential Distributed Objects Survival Guide, New York: John Wiley &Sons, 1996

[3] Krasner G. E. et al., A cookbook for using the Model-View-Controller user interface paradiagm in smalltalk-80, J. of Object-Oriented Programming, 1(3)1988

[4] IBM, VisualAge for Smalltalk: Programmer's guide for building parts for fun and profit, Third Ed. IBM Publication SC34-4906-02, 1997

[5] Daniel T. et al., Visual modelling technique: object technology using visual programming, Addison-Wesley Object Technology Series, 1996

[6] Gamma E. et al., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995

[7] Martin F. et al., UML distilled: applying the standard object modelling language, Addison-Wesley Object Technology Series, 1997

[8] 桑大勇、刘西洋等，面向对象可视化重用构件的语言动态特性研究，西安电子科技大学软件工程研究所研究报告，1997年7月

[9] Buschman F. et al., Pattern-Oriented Software Architecture: A System of Patterns, John Wiley& Sons, 1996

[10] Meyer B., The next software breakthrough, IEEE Computer, 30(7)1997

4)目前已经跨越 Windows95、Windows NT 以及 OS/2 Warp，并将进一步支持 IBM AIX。